AA-J055D-TK

Introduction to DECnet Phase IV

Introduction to DECnet Phase IV

Order No. AA-J055D-TK

November 1983

This document is an overview of the concepts and capabilities of DECnet networks. It defines DECnet terms and describes the network functions that DECnet implementations can perform. Readers are expected to be familiar with operating system concepts.

SUPERSESSION/UPDATE INFORMATION: This is a revised manual. It supersedes the Introduction to DECnet (Order No. AA-J055C-TK).

To order additional copies of this document, contact your local Digital Equipment Corporation Sales Office.

digital equipment corporation · maynard, massachusetts

First Printing, November 1983

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Copyright © 1983 by Digital Equipment Corporation

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
VAX	VMS	SBI
DECnet	IAS	PDT
DATATRIEVE	TRAX	

Distributed and Mid-Range Systems Publications typeset this manual using DIGITAL's TMS-11 Text Management System.

MGTPEALL

Contents

Page

Preface

Part I General Information

Chapter 1 Defining DECnet

1.1 What Is DECnet? 1-1
1.2 DECnet Capabilities 1-3
1.3 How Does DECnet Work? 1-4
1.3.1 Interfaces 1-5
1.3.2 Protocols 1-6
1.4 DECnet Phases 1-7
1.4.1 Phase III 1-7
1.4.2 Phase IV 1-8
1.5 DECnet Uses 1-9

Chapter 2 DECnet Concepts

2.1 Nodes 2-1
2.2 Lines and Circuits 2-3
2.2.1 Transmission Modes 2-4
2.2.2 Data Link Protocols 2-5
2.3 Routing 2-5
2.3.1 Routing Terms 2-6
2.3.2 Other Routing Features 2-8

Chapter 3 DECnet Configurations

3.1 DECnet Environments 3-2
3.1.1 Wide Area Networks 3-2
3.1.2 Local Area Networks 3-4
3.2 Node Characteristics 3-8
3.2.1 Phase III and Phase IV Nodes 3-8
3.2.2 Routing Capabilities 3-9
3.2.3 General Purpose and Dedicated Nodes 3-10
3.3 Line and Circuit Characteristics 3-12
3.4 The Total Picture 3-13

Part II DECnet Capabilities and Functions

Chapter 4 Common Mechanisms in DECnet Functions

4.1 Logical Links 4-1
4.1.1 When Are Logical Links Required? 4-2
4.1.2 How to Create a Logical Link 4-2
4.1.3 Behind the Scenes — DECnet Software and Logical Links 4-3
4.1.4 How the Program Identifies Logical Links 4-4
4.1.5 Multiple Logical Links within a Program 4-5
4.2 Data Types 4-5
4.3 Segment Acknowledgment and Retransmission 4-5
4.4 Flow Control 4-6
4.5 Access Control 4-6

Chapter 5 Task-to-task Communication

5.1 DECnet Task-to-task Calls 5-1
5.2 Requesting a Logical Link 5-2
5.2.1 Building a Connect Block 5-3
5.2.1.1 Object Types and Names 5-4
5.2.1.2 Access Control 5-5
5.2.2 Network Specifications 5-5
5.3 Accepting/Rejecting a Logical Link Request 5-6
5.4 Sending and Receiving Data 5-6
5.4.1 Normal Data 5-6
5.4.2 Interrupt Data 5-7
5.5 Terminating the Link 5-8

Chapter 6 Remote File and Record Access

6.1 Programming Remote File Access 6-2
6.1.1 File System Capabilities 6-3
6.1.2 Initiating Remote Access 6-3
6.1.3 The File Specification 6-4
6.1.4 Access Control Information 6-4
6.1.5 File Characteristics 6-5
6.2 Accessing Remote Files from a Terminal 6-5
6.2.1 Access Control 6-6
6.2.2 File Protection 6-6
6.2.3 Remote File Specifications 6-7
6.2.4 Remote Command File Submission 6-7

Chapter 7 DECnet Terminal Facilities

7.1 Interactive Terminal-to-terminal Communications 7-1
7.2 Network Command Terminal Facilities 7-2

Chapter 8 Internetwork Communications

8.1 X.25 Communications 8-1

8.1.1 DTEs and DCEs 8-2
8.1.2 X.25 Circuits 8-3
8.1.3 X.25 Access Methods 8-4
8.1.4 CCITT Recommendations X.3, X.28, and X.29 8-6

8.2 DECnet/SNA Communications 8-7

8.2.1 DECnet/SNA Circuits 8-9

8.3 Loading the Gateway Node Software 8-9

Chapter 9 How DECnet Supports Applications

9.1 DECnet's Task-to-task Capability in an Application Environment 9-3

9.1.1 Order Entry and Production Scheduling 9-4
9.1.2 Network/Application Interaction 9-4

9.2 DECnet's File Transfer Capability in an Application Environment 9-6

9.2.1 Market Research and Sales Analysis 9-7
9.2.2 Quality Control and Vendor Analysis 9-10

9.3 DECnet's Remote File Access Capability in an Application Environment 9-11

9.3.1 Network Virtual Terminals 9-11
9.3.2 Short-term Cash Management 9-13

9.4 DECnet's Terminal Facilities in an Application Environment 9-13

9.4.1 Order Entry and Production Scheduling 9-14
9.4.2 Market Research 9-14
9.4.3 Short-term Cash Management 9-16

9.5 DECnet as a Consideration in Relocation Planning 9-17

Part III DECnet System Management

Chapter 10 Network System Management

10.1 Network Management Utilities 10-2
10.2 Planning for Node Generation 10-2

10.2.1 Configuration Data Bases 10-3
10.2.2 Network Generation Planning Aid 10-4

10.3 Generating Network Software 10-5
10.4 Defining Configuration and Other Static Parameters 10-5

10.4.1 Node Addresses and Names 10-6
10.4.2 Node Verification Passwords 10-6
10.4.3 Network Object Parameters 10-6
10.4.4 Routing Parameters 10-7
10.4.5 Line Identification 10-7
10.4.6 Circuit Parameters 10-8
10.4.7 Multipoint Line and Circuit Parameters 10-8
10.4.8 Transmission Mode 10-10

10.5 Operating a Node 10–10
10.5.1 Controlling the State of a Node 10–10
10.5.2 Controlling Line or Circuit State 10–11
10.6 Monitoring Node Activity 10–11
10.7 Down-line Loading 10–12
10.7.1 Down-line Loading Definitions 10–13
10.7.2 Down-line Loading Data Base Parameters 10–13
10.7.3 Performing a Down-line Load 10–13
10.7.3.1 The LOAD Command 10–13
10.7.3.2 Target-initiated Down-line Loads 10–14
10.7.4 Down-line Loading and Checkpointing RSX–11S Tasks 10–14

Chapter 11 Monitoring and Testing DECnet Performance

11.1 Integrated Testing Tools 11–2
11.1.1 Loopback Testing 11–2
11.1.2 Trace Capability 11–3
11.1.3 Up-line Dump 11–7
11.2 Monitoring Network Operation with Observer 11–7

Figures

1–1 A DECnet Network 1–2
1–2 ISO and DNA Architectures 1–5
1–3 Protocols and Interfaces 1–7
2–1 Circuits 2–2
2–2 Routing Paths 2–6
2–3 Routing Terms 2–7
3–1 Wide Area Network Configurations 3–3
3–2 A Local Area Network — Ethernet 3–4
3–3 Ethernet Cable 3–5
3–4 Ethernet Hardware 3–5
3–5 DELNI 3–6
3–6 Repeaters 3–7
3–7 Node Routing Capabilities 3–10
3–8 General Purpose and Dedicated Nodes 3–11
3–9 The Total Picture 3–14
4–1 Logical Links and DECnet 4–3
5–1 Addressing Network Objects 5–5
5–2 Transmitting Normal Data 5–7
7–1 Remote Terminal Processing 7–3
8–1 Components of a Packet-switching Data Network 8–3
8–2 Relationships between X.3, X.28, X.29 and the PSDN 8–7
8–3 DECnet/SNA Environments 8–8
9–1 Subsystem and Application Interaction at Fictional Corporation 9–2
9–2 Three Applications Linked by DECnet's Task-to-task Capability 9–5
9–3 Market Research Application Takes Regional and Worldwide Input 9–8
9–4 COPY Request/Response Is Shown Routed through Multiple Nodes 9–9
9–5 File Transfer between Manufacturing and Purchasing Locations 9–10
9–6 Local User Logs onto Remote Node 9–12

9–7 Phone Capability Permits Rapid Communication between Users 9–15
10–1 A Multipoint Line 10–9
10–2 Down-line Load Initiated by a Command Node 10–14
10–3 Down-line Load Initiated by a Target Node 10–15
11–1 Hardware Loopback Devices 11–2
11–2 Command-initiated Loopback Tests. 11–4
11–3 Program-initiated Loopback Tests 11–5
11–4 Line/Circuit Level Loopback Tests 11–6
11–5 Observer Master Menu. 11–9
11–6 Observer Summary Status Display 11–10
11–7 Observer Node Detail Display 11–11

Tables

10–1 DECnet Systems and Network Management Utilities 10–3
10–2 Configuration Data Base Terms. 10–4
11–1 Observer-maintained State and Status Information 11–12
11–2 Statistics Maintained by Observer 11–13

Preface

This revised *Introduction to DECnet* reflects capabilities of the following new products:

- DECnet–VAX Version 3.1 (VMS V3.4)
- DECnet–RSX (RSX–11M and RSX–11S Version 4.0; RSX–11M–PLUS Version 2.0)
- DECnet–20 Version 3.0
- DECnet–10 Version 3.0

In addition, the manual has been reorganized to introduce the following new Phase IV concepts and products.

- PRO/DECnet Version 1.0
- Ethernet and related hardware
- DECnet/SNA Gateway (Version 1.1 for VAX/VMS and Version 1.0 for RSX–11)
- DECnet Terminal Server Version 1.0
- DECnet Router Server Version 1.0
- DECnet Router/X.25 Gateway Version 1.0
- Observer Version 1.1

The Purposes of the *Introduction to DECnet*

Introduction to DECnet describes the concepts and capabilities of DECnet networks. A DECnet network consists of two or more Digital computer systems, enhanced with DECnet software and linked by means of communication lines. DECnet systems in such a network can communicate with each other and share resources. Products are also available that enable DECnet systems to communicate with other network entities, such as IBM SNA mainframes and packet-switching data networks (PSDNs). DECnet networks can be local area networks, wide area networks, or both combined.

Although all implementations of DECnet (for example, DECnet–VAX, PRO/DECnet, DECnet–10, etc.) embody the same network concepts, all do not support the same set of network functions. The specific capabilities of a DECnet network depend on the type of systems participating and on the network's application. The objectives of this *Introduction* are:

- To describe the common network concepts behind all implementations of DECnet
- To identify the major network functions that DECnet provides

Readers of the *Introduction*

The *Introduction* is intended for readers who want to learn about the concepts and capabilities of DECnet systems. It assumes that the reader is familiar with operating system concepts, but not with DECnet. Typical readers will include the personnel at the site of a newly installed DECnet system, who can read this manual to learn about the kind of work DECnet enables them to perform. Another group of readers will include system managers and designers who are thinking about using DECnet to expand their existing Digital computer systems. And, finally, the *Introduction* is also intended for system managers and designers who do not yet use Digital systems, but who are considering the implementation of a computer network. This manual will inform them about the network capabilities that DECnet provides.

The Structure of the *Introduction*

The *Introduction* consists of three parts, encompassing eleven chapters:

- The first part, Chapters 1 to 3, introduces the concepts and general configuration guidelines for DECnet.
- The second part, Chapters 4 to 9, defines specific network functions and explains the mechanisms that programmers and terminal users can employ to implement those functions. Examples of how these capabilities can be used in an application environment are also presented in this part of the book.
- The third part, Chapters 10 and 11, introduces DECnet functions related to the management of the systems that make up a DECnet network.

Associated Documents

This *Introduction* discusses many topics that are explained in greater detail in other manuals. The appendix section of the *Overview of Digital Networking Products* manual contains a complete list of DECnet manuals as well as manuals on other products pertinent to DECnet, such as Ethernet and Packetnet System Interfaces (PSI).

GENERAL INFORMATION

Chapter 1
Defining DECnet

This manual introduces you to DECnet. It tells you what DECnet is, how and where it is used, and what it offers you. The manual provides you with a basic orientation that can be applied to all DECnet products, and it refers you to other manuals that provide more detailed information.

1.1 What Is DECnet?

DECnet is the name given to the family of software and hardware communications products that enable Digital operating systems to participate in a network environment. A network is an aggregate of two or more computer systems — called nodes — connected to each other by physical links (cable, microwave, or satellite links) for the purposes of engaging in communications and resource sharing. Figure 1–1 shows an example of a DECnet network.

A DECnet network can be large, consisting of up to 1023 nodes, or small, consisting of only two nodes. Any processor, intelligent terminal, or other computer system capable of functioning autonomously within a DECnet environment can be a DECnet node. Nodes that are physically located next to each other are called adjacent nodes. Adjacent nodes communicate by means of logical communications paths — called circuits. Circuits operate over the physical links between adjacent nodes to convey all communications and data exchanges that occur in a DECnet network. Information is carried over circuits from node to node, enabling communications to reach any node in the network. Circuits and the way they are used to convey data are explained more fully in Chapter 2.

A very important feature of a DECnet network is that DECnet nodes have a peer relationship with each other. The network does not have to be structured in a hierarchical manner. You can configure a network in which every node can communicate with every other node without consulting a central controlling node. In such a decentralized network, each node can be equally responsive to user requests; network users can gain easy access to applications and facilities that exist on other nodes; and communication overhead can be reduced.

Decentralization also provides a great deal of flexibility in the way that networks can be configured. The type of configurations possible in a decentralized network are discussed in Chapter 3.

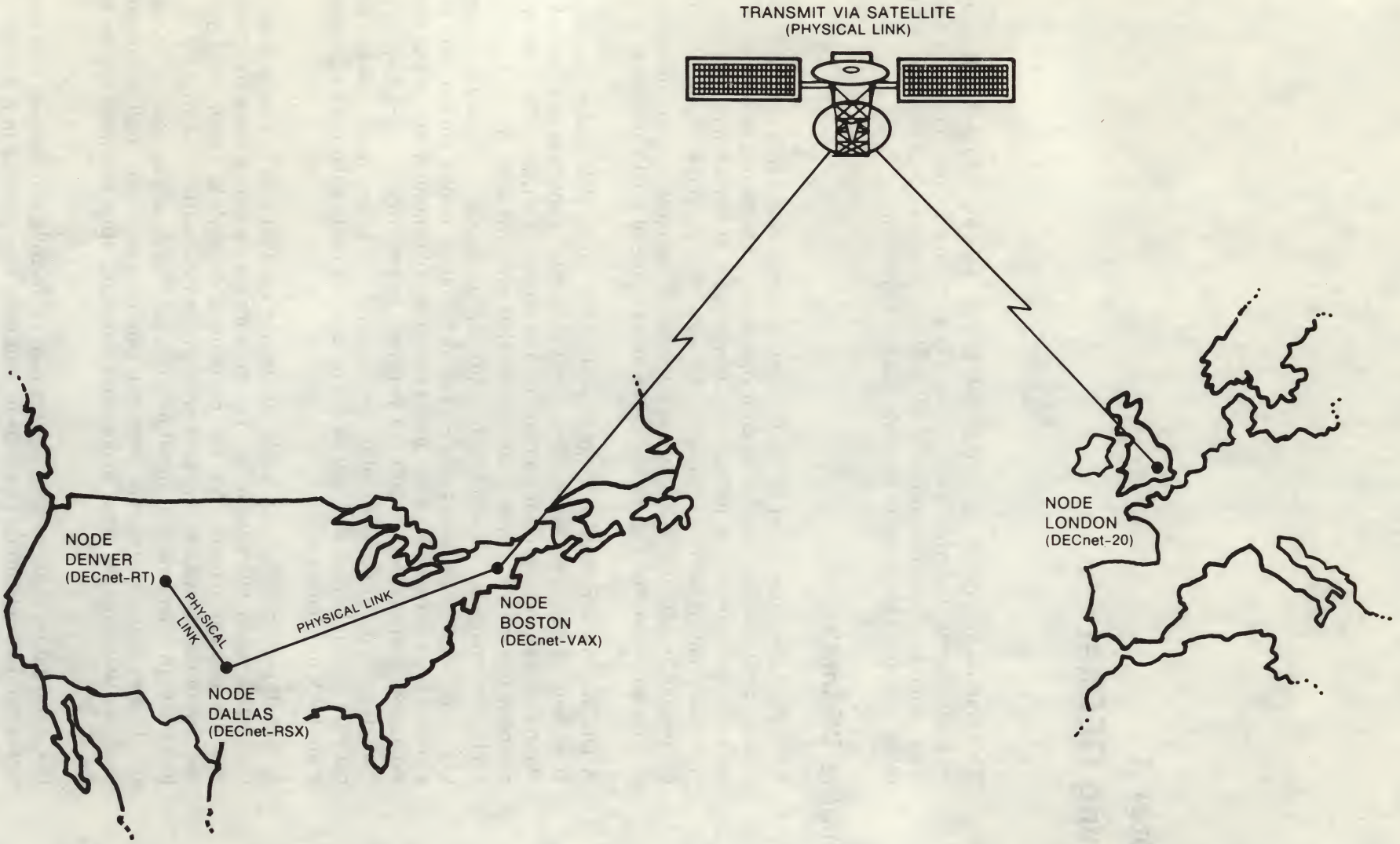


Figure 1–1: A DECnet Network

1.2 DECnet Capabilities

As described in Chapters 4, 5, 6, 7, and 8, DECnet extends the following capabilities to Digital operating systems:

- **Task-to-task communication.** Two programs, on the same or on different network systems can exchange data.
- **Remote file and record access.** DECnet provides both terminal and program access to files that reside on remote nodes. Remote file access facilities allow you to perform the following operations:

 Transfer files between two nodes

 Manipulate files residing on a remote node (for example, open, delete, or append data to remote files)

 Submit command files to a remote node for execution to gain access to that node's resources
- **Terminal-to-terminal communication.** On some DECnet implementations, a terminal user can use a DECnet utility to send messages to other terminals in the network.
- **Network terminal communication.** A local terminal can be connected logically to a remote node. (The remote node executes commands typed at the local terminal.) In Phase III DECnet implementations, the remote node is a DECnet node supporting the same operating system; in Phase IV implementations, the remote node can support a different operating system.
- **Topology alternatives.** DECnet provides the following features, which give you considerable flexibility when configuring a network:

 Ethernet configurations

 An Ethernet configuration provides a local area network environment for DECnet users. Ethernet is a high performance environment and supports a variety of dedicated communications servers as described in Section 3.2.3.

 Adaptive routing

 Any node in a DECnet network can communicate with any other node as long as there is some physical pathway between them. Section 3.2.2 provides additional information on adaptive routing.

 Multipoint lines

 Several DECnet nodes can share a single line in a multipoint configuration. Section 3.3 provides additional information on multipoint lines.
- **Network management.** System managers can use DECnet commands and procedures to generate, define, monitor, and maintain DECnet nodes.

- **Down-line loading.** System images and programs can be down-line loaded to or up-line dumped from certain nodes. For example, an RSX-11S DECnet node which has crashed may up-line dump a copy of its system image to an RSX-11M DECnet node for interpretation.

- **Problem isolation.** System managers can use DECnet commands and procedures to exercise various network capabilities and isolate network problems. Loopback tests, trace facilities, and dump analyzers are some of the procedures available.

- **Access to foreign vendor networks.** DECnet interacts with Digital gateways to enable users to communicate with IBM systems in an SNA network. Gateways are also used for communication with nodes supporting X.25 protocol over a packet-switching data network (PSDN). Information on foreign vendor networks is provided in Chapter 8.

- **Data link independence.** DECnet provides the option of communicating over different data links. Supported data links implement DDCMP (Digital's data communications protocol), X.25 (an international packet-switching protocol) and Ethernet (a local area network protocol). The implementation of any of these data links is for the most part transparent to the user and the applications software. Section 2.2.2 provides additional information on data link protocols.

Chapter 9 contains examples of applications that demonstrate how these capabilities support application interaction.

1.3 How Does DECnet Work?

DECnet nodes are able to communicate effectively by implementing a set of rules that govern how DECnet software components relate to each other. These rules are collectively called the Digital Network Architecture (DNA). As shown in Figure 1-2, DNA rules are similar to those that make up the Reference Model for Open Systems Interconnection developed by the International Standards Organization (ISO).

NOTE

The framework of DNA is transparent for most user activities. You should be aware that DNA exists and that it defines design rules for DECnet software. However, you will most likely have no need to be involved with the details of DNA.

A brief summary of DNA is included here for your convenience. If you require further detail, refer to the manual *DECnet Digital Network Architecture (Phase IV) General Description.*

DNA defines two kinds of relationships between DECnet software components:

1. Interfaces
2. Protocols

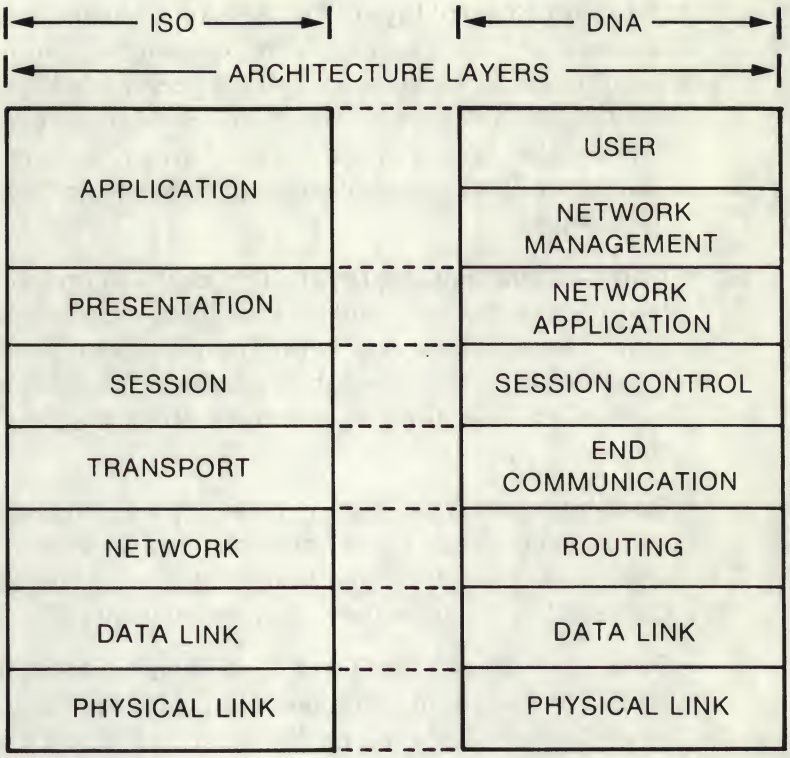


Figure 1–2: ISO and DNA Architectures

1.3.1 Interfaces

Interfaces are specific boundaries that DNA defines between DECnet software components residing within the same node. These boundaries are structured as a set of vertical layers (shown in Figure 1–2). DECnet software is arranged, according to function, as modules within these layers:

- **User layer.** The User layer encompasses user-written programs and user-level services that access the network.
- **Network Management layer.** The Network Management layer software defines the functions used by operators and system programs to plan, control, and maintain the operation of DECnet networks.
- **Network Application layer.** The Network Application layer software defines network functions used by the User and Network Management layers. The most important DECnet functions currently operating in this layer are remote file access, file transfer, remote terminal capability (including the network virtual terminal function), access to X.25 connections using a Packetnet System Interface (PSI) or X.25/X.29 extension package, and access to SNA Gateways. (Chapters 6 through 8 discuss these capabilities.)

- **Session Control layer.** The Session Control layer software defines a mechanism that allows a program in one node to communicate with a program in another node, regardless of either program's location in the network. Modules in the User layer, the Network Management layer, and the Network Application layer can use the mechanism provided by the End Communication layer. This mechanism is called the logical link and is further explained in Chapter 4.

- **End Communication layer.** The End Communication layer software handles the system-independent aspects of communications that use the logical link. These include connection management, data flow control, end-to-end error control, and segmentation/reassembly of user messages. (See the discussion on task-to-task communication in Chapter 5 for clarification of these concepts.)

- **Routing layer.** The Routing layer software defines the mechanism for routing user data from the sending node to the receiving node. Modules in this layer also provide congestion control and packet lifetime control. (See Chapter 2 for clarification of these concepts.)

- **Data Link layer.** The Data Link layer software defines a mechanism for error-free communication between adjacent nodes whether they are connected by an X.25 link, an Ethernet link, or a DDCMP link (point-to-point or multipoint as described in Section 3.3). See Chapter 2 for a discussion of Ethernet and Chapter 8 for a discussion of X.25.

- **Physical Link layer.** The Physical Link layer defines the way that device drivers and communications hardware (modems, lines, etc.) are used to move data over a transmission line.

According to DNA, DECnet software modules in each layer have to interact with each other in a specified way. A module can use services provided by modules in a lower layer and can provide services to a higher layer. However, a module cannot access services offered by a module at a higher layer. In building block fashion, DNA causes each layer of software to build on the services offered by the modules in a lower layer. By adhering to these rules, functions such as the way communications are established between two programs occur in a predictable, regulated way on each DECnet node.

1.3.2 Protocols

Protocols are rules that define the relationship between software modules operating in different nodes. Simply put, protocols provide a common code of behavior that two computer systems in a network can use to understand the role each one is playing in communications and information exchanges.

Protocols define the form and content of messages exchanged by software modules that serve the same network function and reside in the same DNA layer in different nodes. Figure 1–3 shows this relationship.

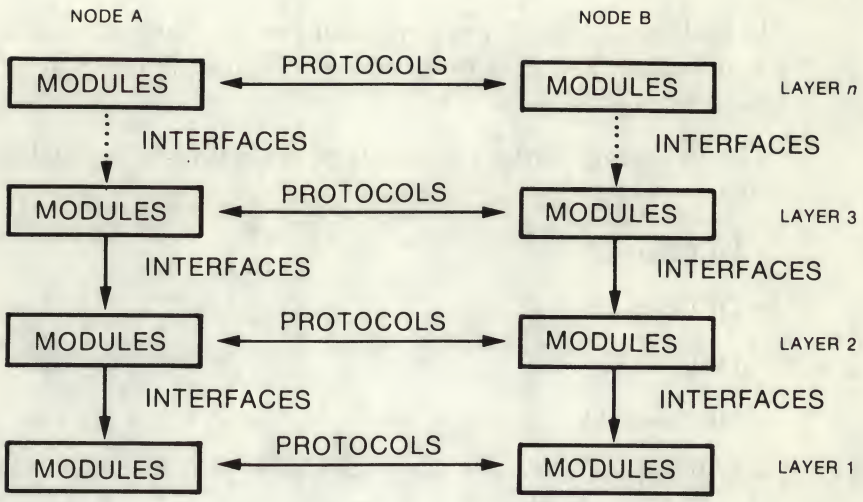


Figure 1–3: Protocols and Interfaces

DNA does not define protocols for all functional layers. In some cases, such as User layer programs that communicate over the network, the programmer must interpret the data received. (This is similar to interpreting the contents of a record on disk.) Furthermore, multiple protocols can be defined for the same layer, because some layers support more than one function. Also, the protocols that DNA does define are not exclusive: users can substitute their own protocols, if they choose, as long as they are implemented consistently by equivalent modules throughout the network.

1.4 DECnet Phases

Digital has developed networking products based on the DNA architecture since 1975. This development effort has occurred in a series of phases. To date, there have been four DECnet phases. The products in each new phase are fully compatible with those produced in the previous phase. Sections 1.4.1 and 1.4.2 describe capabilities of Phase III and Phase IV DECnet nodes. (Appendix B in the *Overview of Digital Networking Products* contains a brief history of DECnet and discussions of earlier DECnet phases.)

1.4.1 Phase III

DECnet Phase III systems can perform the following functions (see Section 1.2):

- Task-to-task communication
- Remote file access
- Terminal-to-terminal communication
- Network terminal communication
- Network management
- Down-line loading
- Loopback testing

In addition, Phase III systems can perform message routing, multipoint communication, and communications with packet-switching data networks using X.25 protocols.

The following DECnet products provide Phase III capabilities as discussed in this manual.

- DECnet-20
- DECnet-10
- DECnet/E
- DECnet-RT
- DECnet-IAS

1.4.2 Phase IV

Phase IV added technology that established Digital as a leading supplier of products for multivendor network environments. In addition to being backward compatible with Phase III, Phase IV features include:

- Data link independence which provides expanded data link options to DECnet users. Data links supported by DECnet are X.25, DDCMP, and Ethernet links.
- Dedicated communications server products that off-load communications overhead from general purpose nodes (see Chapter 3).
- Increased network routing support allowing up to 1023 nodes to be supported in a network.
- Enhanced network management capabilities (see Chapters 10 and 11).
- Gateway support for communications with nodes using X.25 protocol over a packet-switching data network (see Chapter 8).
- Gateway support for communications with IBM systems in an SNA (System Network Architecture) network (see Chapter 8).

The following DECnet products implement Phase IV capabilities as discussed in this manual:

- DECnet-RSX (11M, 11M-PLUS, and 11S)
- DECnet-VAX
- PRO/DECnet

1.5 DECnet Uses

Some of the advantages of a network over a single system or group of unconnected single systems is that you can use it to increase the flexibility of your system and the way you perform certain tasks. Consider the following ways in which you might use DECnet:

- **Resource sharing.** In a network environment, you do not need to duplicate resources for each node. Line printers, storage facilities, processing capabilities and dedicated communications servers are among some of the resources that can be shared by multiple nodes.
- **Remote file access.** Having access to files on remote network nodes expands the options you have as to where you create and store information. You can maintain fewer copies of a given file and users can always access the most current version.
- **Distributed data bases.** Any network node can access information stored on any other network node. This means that users throughout the network can make use of information stored in common data bases.
- **File transfer.** Multiple users in a network can exchange files rapidly. This capability speeds up the information flow between organizations and decreases the amount of paperwork that must be handled.
- **Communication with foreign vendor systems.** DECnet interacts with Digital gateways and special software products to enable you to communicate with non-DECnet systems. You can communicate with IBM SNA systems using a DECnet/SNA Gateway. Using a DECnet Router/X.25 Gateway and/or related software, you can communicate with any node connected to a packet-switching data network that implements the appropriate X.25 protocols. This expands network access and resource potential for communications over a range of different networks.

Chapter 9 of this manual discusses other applications of DECnet capabilities.

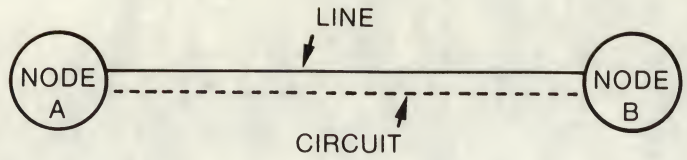
Chapter 2
DECnet Concepts

Network activity involves the flow of information from one node to another. To understand how this happens in a DECnet network, there are some key DECnet concepts you should know. These concepts are described in this chapter in the following order:

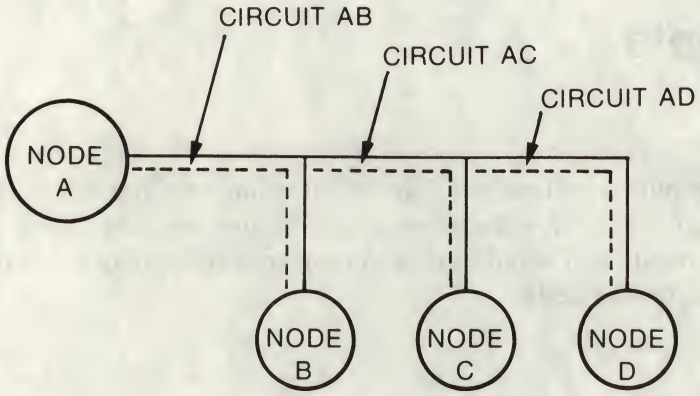
- Nodes
- Lines and circuits
- Routing

2.1 Nodes

A node is a network entity — an identifiable unit capable of processing, sending, and receiving network information. A node consists of an autonomous software entity such as an operating system, with processors and other associated hardware implementing DECnet products to perform network operations. Every node in a DECnet network has a unique numeric address. This address designates the location of a node in the network just as a street address designates the location of a building in a town. When a packet of data is sent from one node to another, it is enveloped by control information, which includes the address of the sending node (called the source address) and the address of the receiving node (called the destination address). This information tells nodes that intercept the packet en route to its destination where the packet is going. Node addresses ensure that a data packet can always be identified in terms of who is sending it and who is receiving it, no matter where in the network it travels. (See Section 2.3 for a discussion of the routing mechanism that determines data packet paths.)



POINT-TO-POINT CONNECTION



MULTIPOINT CONNECTIONS

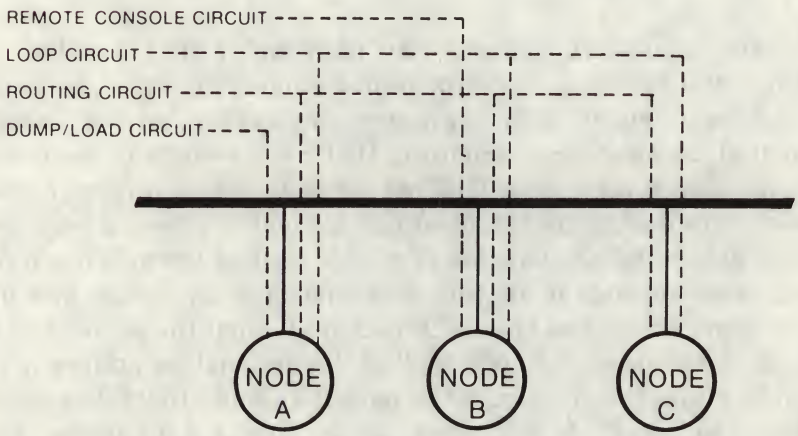


Figure 2–1: Circuits

For most users, however, addresses are difficult to remember, particularly when you are dealing with a large number of nodes. Therefore, nodes can be assigned node names, which are easier to remember. The system manager for each node specifies names for all the nodes with which that node expects to communicate. These names are defined in the local data base and are meaningful only to the local software. Each system manager can define a different node name for the same remote node. (DECnet software in each node keeps a list containing the node address associated with each node name. The node address defined for each node must be consistent throughout the network.) This allows users to refer to nodes by name and the software to find the correct address of a node in order to route data properly.

The node name and address by which a node is to be known in the network are usually assigned by the system manager when a node is initially generated or configured into the network. For small networks, these assignments can be hand-typed into each system; for large networks, it is recommended that an up-to-date master list of node addresses and names be maintained on a particular node. That way, other nodes in the network can simply copy the information from the master file. (Existing nodes in the network exchange their node names and addresses with each newly configured node.)

Refer to the system manager's guide for your system for more information on specifying node names and addresses. Other node characteristics, such as routing capabilities and dedicated functions, are discussed in Chapter 3, as they apply to network configurations.

2.2 Lines and Circuits

As noted in Chapter 1, a line is a physical data path from a DECnet node to another DECnet node, or from a gateway to an IBM SNA network or packet-switching data network (PSDN). A line is connected to a node by a hardware controller that drives the line. Lines that use the public telephone network are also often connected to a modem. Lines can be either dedicated (hard-wired) or dial-up. Lines can be attached to a local area network cable or configured in point-to-point or multipoint configurations (as described in Chapter 3).

Whereas lines form the physical connections between adjacent nodes, there is a higher level concept of internodal connections called circuits. Circuits are logical connections that carry information between two nodes and operate over the physical medium of lines. Each point-to-point line has one circuit associated with it and each tributary on a multipoint line has one (as shown in Figure 2–1).

Ethernet uses one physical line with multiple circuits. Each circuit corresponds to a particular Ethernet protocol type. An Ethernet protocol type identifies the type of operation performed over each circuit. For example, the following protocol types are shown in Figure 2–1:

- **DNA routing protocol.** Enables the DECnet routing mechanism (see Section 2.3) and is required for all DECnet communications.
- **DNA remote console protocol.** Provides simple internodal terminal communications used for diagnostic and maintenance procedures.
- **DNA dump/load protocol.** Enables you to load nodes that do not have mass storage (for example, communication servers).
- **Loopback protocol.** Enables you to perform loop tests and is required for any type of testing procedures (see Chapter 11).

There are other protocol types not shown in Figure 2–1. Refer to the bibliography in the *Overview of Digital Networking Products* for a list of Ethernet documentation that further explains protocol types.

It is important to note that all communication activity in the network is considered in terms of circuits. A succession of circuits between a number of intervening nodes can be established by the routing mechanism described in Section 2.3 to get data from one node to another. These circuit pathways enable nodes to send data to other nodes that are not physically adjacent to them.

2.2.1 Transmission Modes

Information is transmitted over DECnet lines in half duplex or full duplex mode.

Half duplex

The line can transmit data in either direction, but only in one direction at any given time. In other words, data cannot be sent and received over the line simultaneously.

Full duplex

The line can transmit data in both directions simultaneously. Full duplex allows a node to send and receive data at the same time.

Half duplex and full duplex relate to the physical line only. Information flow at the higher level of the circuit always appears to be full duplex. The user sends data and it is handled by various layers of DECnet software until it reaches the routing mechanism (see Section 2.3). The routing mechanism determines the circuit over which the data will be transmitted to an adjacent node according to precomputed information. The data is then passed on to software that handles various data link protocols (such as those described in the following section).

As far as the user and the routing mechanism are concerned, the data has been transmitted over the specified circuit. However, if the data cannot be transmitted over the designated physical line because it is a half duplex line already carrying data, or in the case of an Ethernet transmission, because the channel is not free, DECnet software will buffer the data until transmission can be accomplished. The user is not aware of when the data is actually buffered or when it is sent out directly on a free line.

2.2.2 Data Link Protocols

DECnet uses two basic protocols: Digital Data Communications Message Protocol (DDCMP), which is a byte-oriented protocol, and Ethernet protocol with physical channel encoding.

A byte-oriented protocol provides a count of the number of bytes that will be sent in the data portion of each message. The receiver of the message can compare the number of bytes actually received to the number sent by the protocol as a means of error detection. Circuits established between two adjacent DECnet nodes use DDCMP.

Physical channel encoding uses the carrier signal to indicate the beginning and end of a message. When a signal is detected by the intended receiver of a message, the message has begun transmission. When the signal ends and clock transition is no longer detected, the message has ended. Ethernet circuits use the Ethernet protocol with Manchester physical channel encoding. Refer to the *Ethernet Products and Services Catalog* for detailed information.

NOTE

See Chapter 8 for a discussion of X.25 and DECnet/SNA data links and circuits.

2.3 Routing

Routing is the network function that determines the path or "route" along which a packet of data travels to its destination. A path consists of the sequence of circuits and nodes located between the source node and the destination node. In many cases, there are multiple paths between network nodes (as shown in Figure 2–2).

How do you know over which path to send data? What do you do if a line on the path goes down? Fortunately, DECnet software implements a routing mechanism on each node that handles that worry for you.

You simply address and transmit data according to the type of function you are using (user program, terminal utility, and so on). The DECnet routing mechanism deciphers the destination address of the packet and forwards it along to the destination node according to a precomputed path that may include intervening nodes. (The path is computed by the routing function according to information on path cost and distance that is kept in a routing data base in each DECnet node. This information is updated dynamically if conditions in the network change. For example, if a line goes down, nodes affected by that line would recompute their routing information to direct data traffic over an alternate path.)

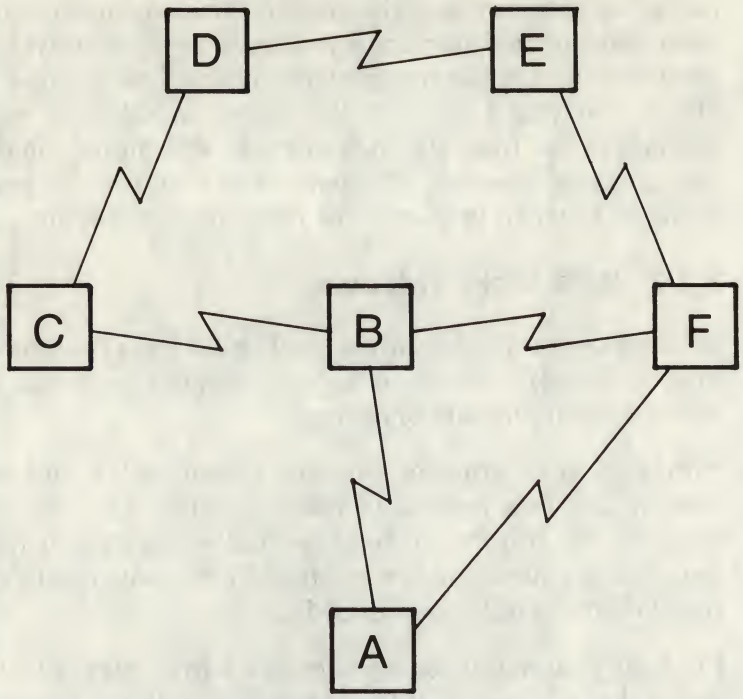


Figure 2–2: Routing Paths

When a message is sent to an Ethernet node from a node not connected to the Ethernet, it must be received by a routing node. The routing node submits the message to the Ethernet channel. The message travels along the Ethernet channel until it reaches the destination node. (The destination node can recognize and retrieve messages addressed to it.)

If a message originates on an Ethernet and is addressed to another node on the same Ethernet, a routing node is not necessary. The sender transmits the message directly to the destination as if they had a point-to-point connection.

NOTE

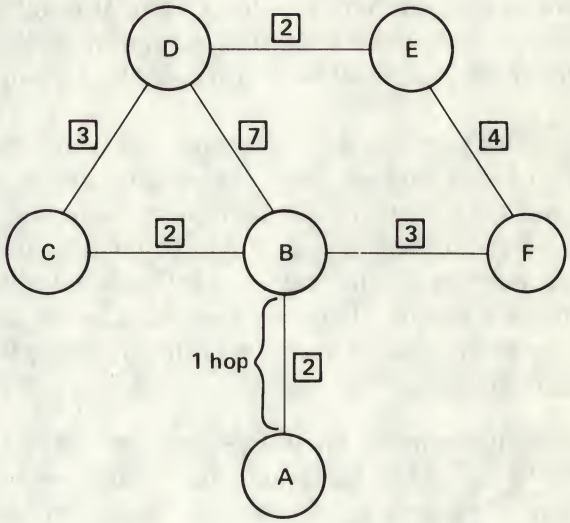
If you are implementing X.25 protocol through a gateway node rather than through data link mapping as discussed in Chapter 8, DECnet routing is not used over the PSDN. The PSDN handles all routing functions that occur within its boundaries. (See Chapter 8 for an explanation of X.25 implementations.)

2.3.1 Routing Terms

The total distance between a source node and its destination is called the path length. The path length is measured in hops. A hop is equal to a circuit between two nodes. A path never exceeds a maximum number of hops, which is a value set by a system manager or determined by the DECnet implementation.

The cost of each path between a source node and a destination node is determined by the sum of positive integer values assigned to the circuits that compose the path. Each integer value is called a circuit cost.

When generating a network data base, a system manager or operator assigns a cost to each circuit defined for that node. When the node is up and running, an operator can dynamically change individual costs to higher or lower values. Altering circuit costs can change packet-routing paths. Figure 2–3 shows the correlation between path length, hops, and path cost.



Legend:

(X) = node

——— = circuit

[n] = circuit cost

(X)—(X) = hop

Node A wants to send a packet to Node D. There are three possible paths.		
PATH	PATH COST	PATH LENGTH
(A) to (B), (B) to (C), (C) to (D)	[2] + [2] + [3] = 7*	3 hops
(A) to (B), (B) to (D)	[2] + [7] = 9	2 hops
(A) to (B), (B) to (F), (F) to (E), (E) to (D)	[2] + [3] + [4] + [2] = 11	4 hops

*7 is the lowest path cost; Node A therefore routes the packet to Node D via this path.

Figure 2–3: Routing Terms

2.3.2 Other Routing Features

In addition to determining path length and path cost, the routing mechanism also does the following things:

- Limits the number of packets queuing up for transmission on individual circuits.
- Regulates the ratio of packets to be forwarded through a node with those that are generated on the node, to achieve a reasonable balance of traffic from both sources.
- Tracks the number of nodes a route-through packet has visited and discards packets that have exceeded a predefined limit. This ensures that packets can never loop endlessly through the network.

Although the routing mechanism has been designed to operate without need of direct user intervention, system managers and operators can exercise some control over routing performance by manipulating certain configuration parameters such as maximum path lengths and circuit costs. A system manager can define initial values for these parameters when building a DECnet system for a node. These values can also be modified after a DECnet system has been installed to improve performance or to reflect changes in the network configuration.

Routing parameters should be determined only after careful consideration of their effects on the local node and on the network as a whole. See the DECnet system manager's guide for your particular system for information on which routing parameters you can specify and how to specify them.

Chapter 3
DECnet Configurations

Understanding the DECnet concepts presented in Chapter 2 will help you to understand how nodes and lines can be arranged in various configurations.

As discussed in Chapter 2, the routing mechanism enables nodes that are not directly connected to each other by physical lines to communicate. This allows great flexibility in the way that nodes and lines can be combined in networks. The major factors that determine how you can join these components are:

- The requirements of the tasks to be performed
- The capabilities of the network components

Scenarios related to the first factor are discussed in the *Overview of Digital Networking Products*. These scenarios illustrate how network components can be mixed and combined in configurations that suit individual needs. You may want to refer to these examples when defining the requirements of your specific tasks.

The capabilities of the components will dictate the bounds within which you can create network configurations to meet your needs. These capabilities are discussed in this chapter as follows:

- **The environment.** There are two basic environments in which you can create DECnet configurations: wide area networks and local area networks. You can combine these environments (see Figure 3.9). Each environment supports different types of configurations.
- **Node characteristics.** Certain characteristics of a node, such as its ability to receive and forward messages intended for other nodes and the type of network services it can perform, greatly affect how you position it in a network configuration. Whether the node implements DNA Phase III or Phase IV software is also an important consideration.

- **Line and circuit characteristics.** Lines and their associated circuits can be arranged in configurations such as bus structures, point-to-point arrangements, or multipoint arrangements.

NOTE

This chapter on configurations is not intended to describe configuration programs for specific DECnet systems or to give you definite specifications for configuring your network. Rather, it is a general discussion on the types of configurations that are possible using DECnet. For specific information on how to configure a system into the network, refer to the system manager's guide for your particular DECnet system.

3.1 DECnet Environments

There are two basic environments for DECnet configurations: wide area networks and local area networks (LANs). Figure 3–1 shows an example of a wide area network, and Figure 3–2 shows an example of a local area network.

Each type of network can be configured with a variety of nodes and links, as defined in Sections 3.2 and 3.3. However, only Phase IV DECnet nodes can be directly connected to a local area network cable.

The distinctions between these networks are most important in terms of configurations. The end user does not see any apparent difference between communications with nodes on a wide area network and those on a local area network except that performance improves over the local area network.

3.1.1 Wide Area Networks

A wide area network is composed of nodes connected by individual communications links configured in various patterns as shown in Figure 3–1.

These networks are used to transfer data over distances ranging from a few feet to thousands of miles. This disparity in distance occurs because (a) wide area networks are always used for long distance communications, (b) there are some short-distance connections, such as a point-to-point connection between two nodes or a connection between time-critical nodes, where it is impractical to set up a local area network. Such short distance communications are better handled using high speed coaxial cable and other traditional communications hardware.

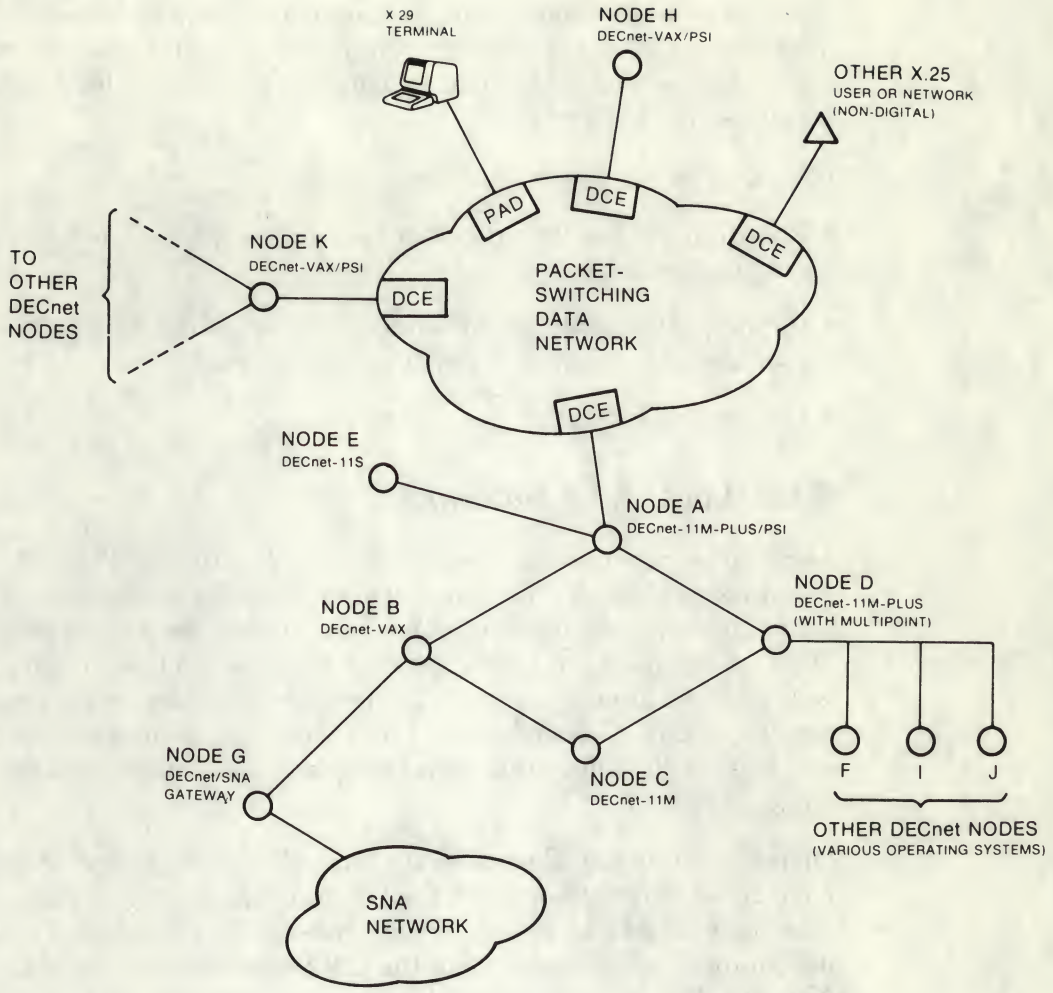


Figure 3–1: Wide Area Network Configurations

Messages can be transported in a wide area network over leased or switched lines. Typically, wide area networks use common carriers, such as the telephone network, to transport messages over most or part of these distances. Because wide area networks use the telephone system for most communications, modems or modem equivalents are generally needed on each end of a communications link to convert digital data to analog form and to permit transmission over long distances.

Different transmission speeds can be specified for different communications lines in a wide area network. Transmission speeds depend on the supported hardware devices and the type of lines used. This allows some leeway in distributing work loads and adjusting performance levels for various network components.

You can combine nodes in wide area configurations according to the guidelines discussed in Section 3.2. Any restrictions to wide area node configurations, such as nodes that can be connected only to a local area network, are identified in that section.

Examples of wide area networks include:

- Private packet-switched networks, such as the data networks developed by many companies
- Public packet-switched networks, such as public data networks using X.25 protocol (for example, Telenet and Transpac)
- Leased line networks

3.1.2 Local Area Networks

Local area networks are privately owned networks that offer reliable high speed communication channels. These networks are optimized for connecting information-processing equipment in a limited geographic area — namely, an office, a building, or a complex of buildings, such as a campus. These networks can be designed with a variety of technologies and arranged in different configurations. Consequently, they vary with respect to their transmission speeds, the distances they can span, and the capabilities and services they offer.

Digital Equipment Corporation, Intel Corporation, and Xerox Corporation have collaborated to produce the Ethernet specification, which defines a local area network having specific characteristics (see Figure 3–2). Digital's Ethernet protocol is implemented in the DNA architecture (see the *DECnet Digital Network Architecture Phase IV General Description* and the Institute of Electrical and Electronics Engineers (IEEE) 802.3 draft standard for details).

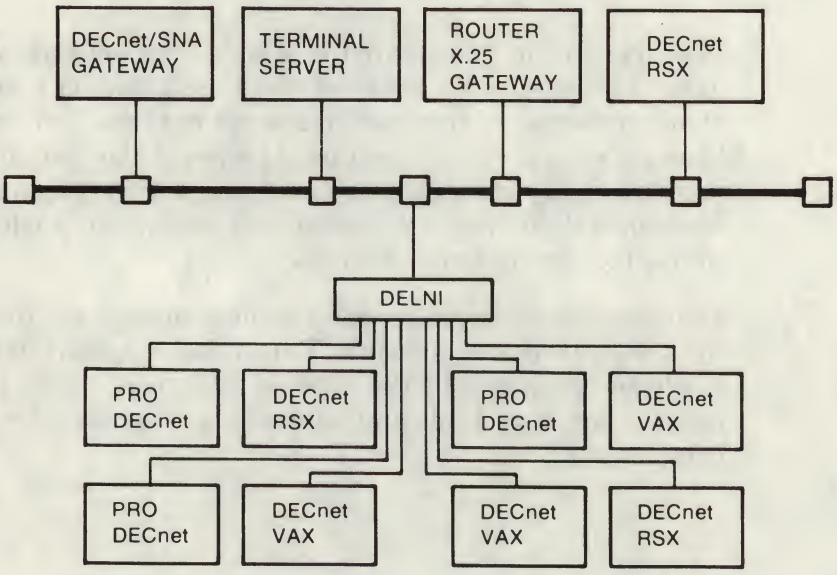


Figure 3–2: A Local Area Network — Ethernet

Ethernet is characterized by one or several segments of coaxial cable joined together, each segment ranging from 20.4 to 500 meters long. The Ethernet cable is terminated at both ends (as shown in Figure 3–3).



Figure 3–3: Ethernet Cable

A Phase IV Ethernet network supports up to 1023 nodes. Ethernet supports a bus topology, with each node attached to the cable by a single line. The channel bandwidth of an Ethernet is 10MB with a CPU throughput for each controller set at a maximum of 1.3MB. This is considerably faster than conventional configurations.

Nodes that support a UNIBUS structure, such as PDP–11 and VAX nodes, connect to Ethernet by means of a network adapter called the DEUNA; nodes with a Q–BUS structure, such as micro-11 systems, connect using a network adapter called the DEQNA. The DEUNA and the DEQNA each incorporate hardware module boards, a distribution panel, and cabling that is installed as a communications device in a DECnet node. These adapters connect to the H4000 transceiver physically and electrically by a transceiver cable (as shown in Figure 3–4).

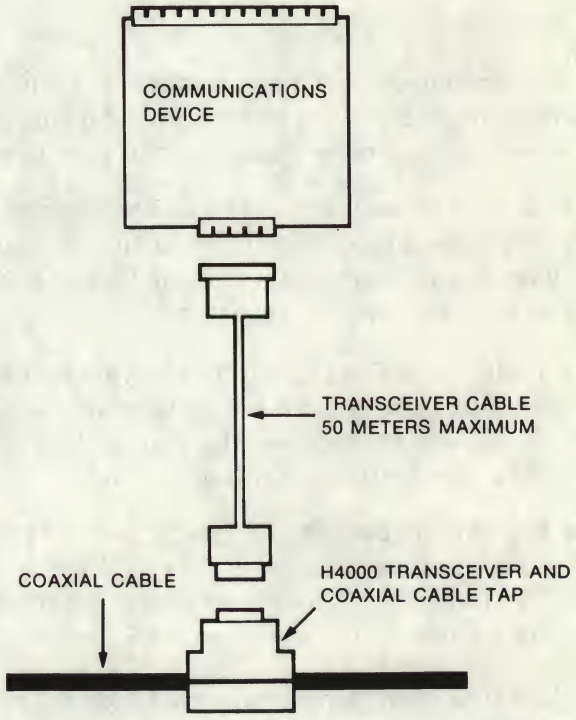


Figure 3–4: Ethernet Hardware

A maximum of 100 transceivers can be used on a single 500 meter segment of Ethernet cable. A communications controller called the DELNI allows you to connect up to eight nodes or devices to each H4000 transceiver. (You can also use the DELNI apart from the Ethernet. In this case, the DELNI and connected devices form a local area network of their own. When not connected to an Ethernet, DELNIs can be nested 1 deep to provide 64 connections.) Figure 3–5 shows PRO/DECnet nodes on a DELNI attached to an Ethernet cable.

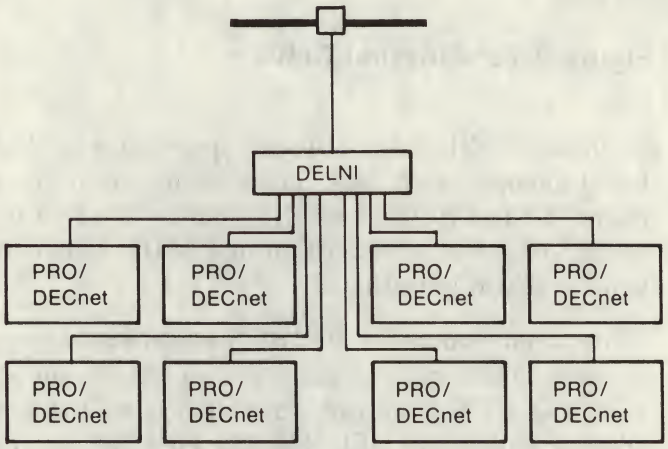


Figure 3–5: DELNI

As mentioned previously, segments of cable can be connected to extend Ethernet beyond the 500 meter limit of a single segment. Two cable segments are joined using a device called a repeater (see Figure 3–6).

A repeater enables the two cable segments to function as if they were one cable. It amplifies transmission signals and passes data packets between two coaxial cable segments without filtering signals. Digital local area networks can use two types of repeaters:

- **Local repeater.** Connects two coaxial cable segments within a limited geographic distance, for example, within a building. There can be a maximum of 100 meters between the two cable segments and each cable segment can be a maximum of 500 meters long.
- **Remote repeater.** Connects two coaxial cable segments spanning a distance greater than that covered by a local repeater, for example, between two buildings. The remote repeater consists of two local repeaters connected by a fiber optic link up to 1000 meters (3282 feet) long.

Both local and remote repeaters connect to an H4000 transceiver on the coaxial cable using transceiver cables (see Figure 3–6). No more than two repeaters can be placed between any two nodes on the Ethernet.

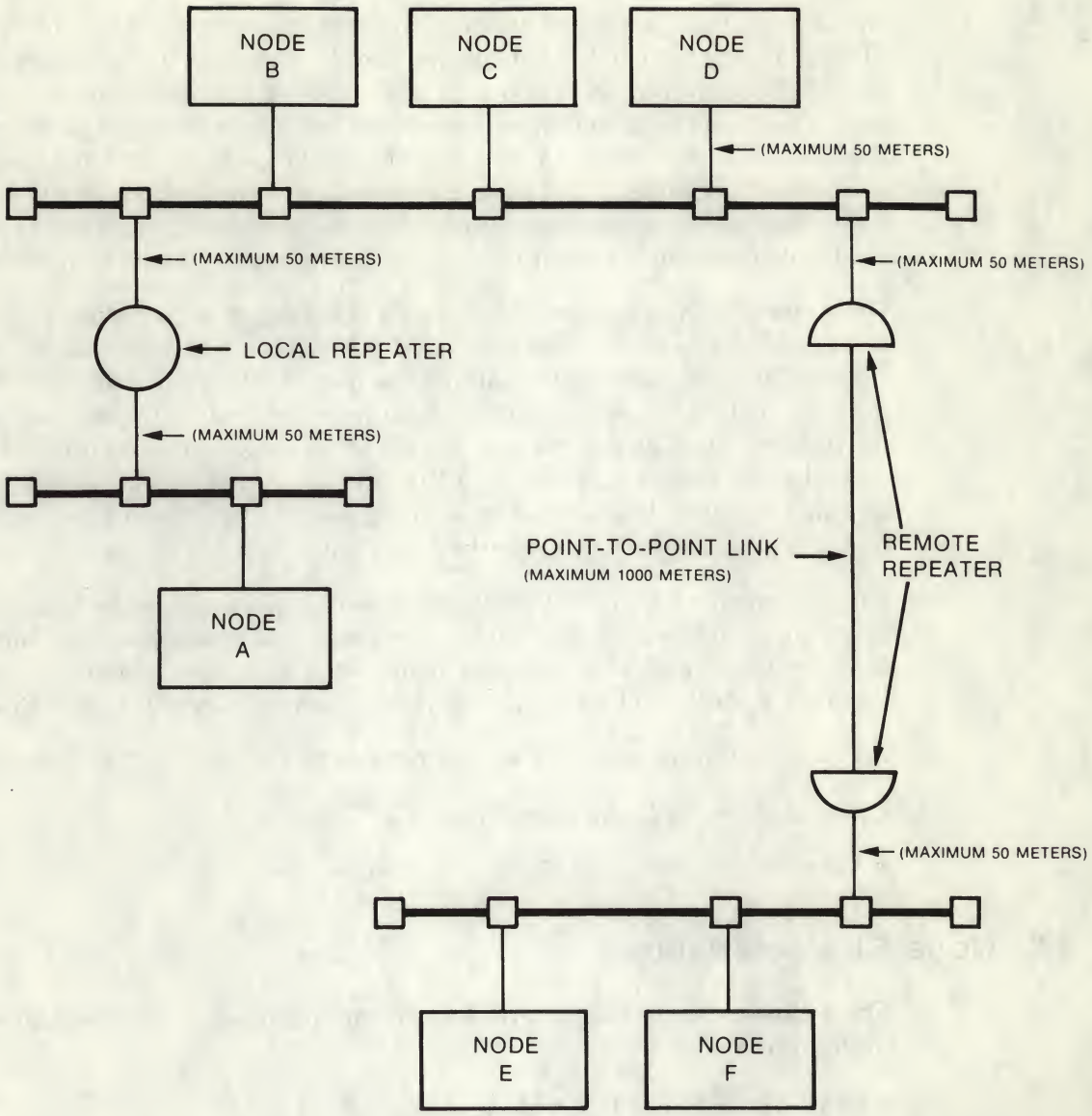


Figure 3–6: Repeaters

Other characteristics of Ethernet configurations are:

- Transceiver cables connecting nodes to the Ethernet cannot exceed 50 meters.
- Cable segments must be terminated in 50 ohms.
- The maximum distance between any two nodes is 2800 meters. (This includes the length of the Ethernet cable, the transceiver cable, a local repeater, and a remote repeater.)

All DECnet nodes directly connected to an Ethernet cable must be Phase IV DECnet nodes. However, Phase III nodes or Phase IV nodes not on the Ethernet can gain access to Ethernet resources through a DECnet Router Server, a DECnet Router/X.25 Gateway, or a full-function Ethernet node.

All server nodes described in Section 3.2.3 are supported on the Ethernet. (There is a DECnet/SNA Gateway implementation that supports DECnet/SNA communications in a wide area network configuration as well.) Each server node on the Ethernet is down-line loaded by a designated host node on the same Ethernet cable. A host node is a full capability DECnet node whose services are required in the start-up procedures or operations of another node, as in the case of down-line loading. Documentation for each server node includes installation information describing the requirements of a host node.

One type of DECnet system, called PRO/DECnet, can participate in network activities only as an Ethernet node. (PRO/DECnet software enables a Professional 300 series personal computer to participate as a node in a DECnet network. PRO/DECnet is distinguished by a menu-driven user interface. Refer to the PRO/DECnet manual set for more specific information.) Once attached to an Ethernet, however, PRO/DECnet can access wide area resources as can any other DECnet node — using routers or other Phase IV DECnet nodes that span the two networks (see Figure 3–9).

Nodes connected to an Ethernet cable can connect with wide area networks, other local area networks, packet-switched data networks that implement X.25 protocol, and IBM systems using SNA protocol. In fact, as shown in Figure 3–9, X.25 and SNA Gateways can connect directly to the Ethernet.

For more information on Ethernet, refer to the following Digital manuals:

- *Introduction to Local Area Networks*
- *Ethernet Products and Services Catalog*

3.2 Node Characteristics

The following node characteristics are important considerations in network configurations:

- Phase III or Phase IV DNA implementation
- Routing capabilities
- Dedicated or general purpose function

These characteristics are discussed in the three subsections that follow.

3.2.1 Phase III and Phase IV Nodes

As mentioned in Chapter 1 of this manual, DECnet products have been developed in a series of phases. Not all Digital operating systems implement the same phase of DECnet, nor do they all implement the full capabilities of each phase. Chapter 1 discusses the DECnet phases and capabilities currently supported by each Digital operating system.

In planning a network configuration, the phase of DECnet that is installed on a node is important, since Phase IV DECnet extends the networking capabilities of a node beyond what is offered in Phase III. Phase IV capabilities that affect configurations are as follows:

- Only Phase IV nodes can directly connect to an Ethernet cable. Phase III nodes can communicate with Ethernet nodes through a router server or a Phase IV node.
- The Phase IV routing mechanism supports network communications involving up to 1023 nodes, while Phase III supports communications for up to 255 nodes.
- Communications servers, described in Section 3.2.3, all implement Phase IV software. Servers are important to configuration plans because they offload functions, such as routing decisions (Chapter 2) and virtual terminal capabilities (Chapter 7), normally performed by a general function node. In an Ethernet environment, for example, you could install a number of nodes without routing software (end nodes) if you have a DECnet Router Server. (Nodes without routing software are less expensive than nodes with it.) The router would provide routing functions for these end nodes.

Throughout this manual, Phase III and Phase IV distinctions will be identified and discussed as they affect various aspects of network operations.

3.2.2 Routing Capabilities

There are two labels applied to DECnet nodes to characterize their routing capabilities:

- End node
- Full-function node

End node

End nodes do not have route-through capability. This means that they cannot receive and forward messages intended for other nodes. An end node can send packets to an adjacent node. If an end node has multiple circuits to one or to several adjacent nodes, only one of those circuits can be active at a time.

If a node adjacent to an end node is a full-function node, its route-through facilities can be used by the end node to establish connections with other nodes in the network. Nodes A, D, and H in Figure 3–7 are end nodes.

Full-function node

Full-function nodes can have multiple circuits actively communicating with one or several nodes at the same time. Full-function nodes also maintain an up-to-date map of the network in their data base. The map provides information, such as the node address of each accessible node in the network. This information is used by the routing mechanism to determine the most cost-effective paths for sending data to a destination node. Data destined for a node that is not adjacent to the sending node can be routed though one or more full-function nodes. Nodes B, C, E, F, and G in Figure 3–7 are full-function nodes.

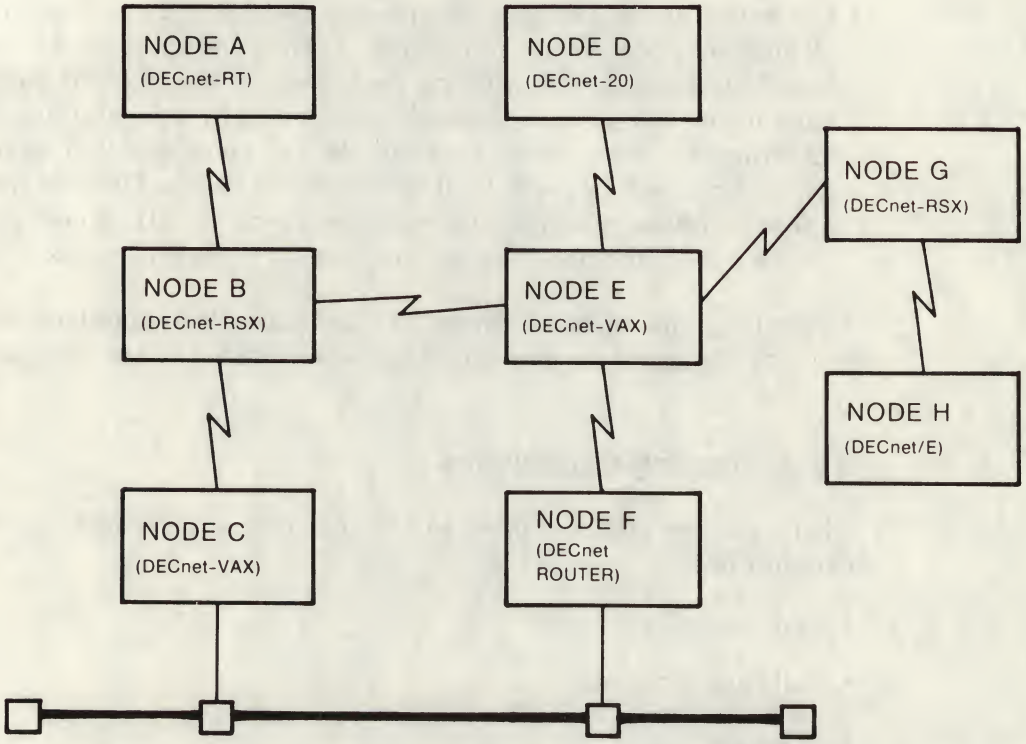


Figure 3–7: Node Routing Capabilities

Chapter 2 describes the routing mechanism and how it determines the communication paths through a network.

3.2.3 General Purpose and Dedicated Nodes

Another factor that affects the way networks are configured is whether or not particular nodes in the network are general function or dedicated purpose nodes. Dedicated nodes have specific functions that can service the needs of many other nodes. For example, as shown in Figure 3–8, some dedicated DECnet nodes provide routing services to a group of nodes on an Ethernet cable, others act as gateways to other networks. Most of the dedicated nodes discussed in this manual must be configured on an Ethernet cable. The one exception is an implementation of the DECnet/SNA Gateway, which can also be configured in a wide area network. General purpose nodes can be configured in a wide area or local area network environment.

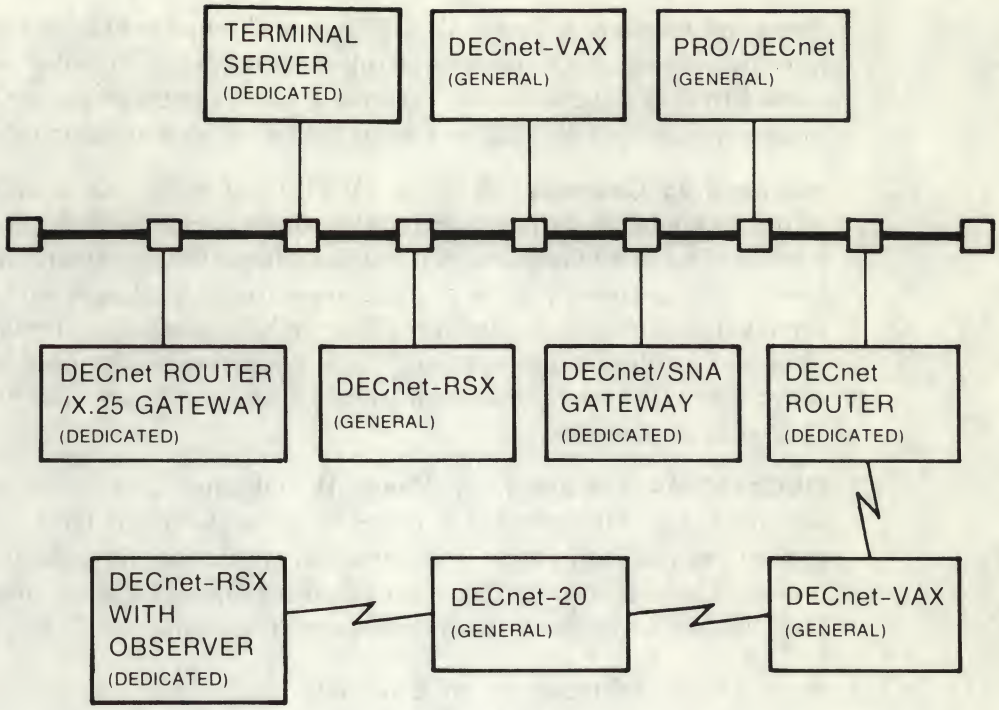


Figure 3–8: General Purpose and Dedicated Nodes

DECnet nodes performing dedicated communications functions are called communications servers. The communications servers discussed here implement Phase IV DECnet software. (Because these servers are dedicated to a specific function, they cannot be accessed as regular Phase IV nodes for general user functions.)

DECnet communications servers must be down-line loaded by a DECnet host node (Phase IV) on the Ethernet cable. (The exception to this is the DECnet/SNA Gateway. A DECnet/SNA Gateway can be attached to an Ethernet and down-line loaded, or it can be located in a wide area network and loaded directly from floppy disk.) DECnet communications servers perform the following functions:

Router Server. A Phase IV DECnet node that performs routing functions only. The router server connects with an Ethernet cable on one side and with other communications servers, Phase III nodes, or Phase IV nodes on the other side.

Routers provide a way through which nodes on and off the Ethernet cable can communicate with each other. Nodes in wide area networks or other local area networks can also communicate with Ethernet nodes by sending messages through a full-function Phase IV DECnet node connected to the Ethernet cable. (In each case, access is provided through the routing mechanism as described in Section 2.3.) You can also use virtual terminal facilities as described in Chapter 7 to form connections between Ethernet and non-Ethernet nodes.

Terminal Server. A Phase IV DECnet node that enables a number of terminals to communicate with multiple host nodes. Terminal servers are connected to an Ethernet cable. Terminals using the terminal server can communicate with Phase IV DECnet hosts on or off an Ethernet cable.

Router/X.25 Gateway. A Phase IV DECnet node that combines all of the routing capabilities discussed for the router server with X.25 Gateway capabilities. The X.25 Gateway capabilities allow this server to connect a DECnet node running specific X.25-compatible software packages with other nodes on a packet-switched data network. Router/X.25 Gateways are connected to the Ethernet cable on one side and to a packet-switched data network on the other side. However, wide area network nodes can also use the services of a Router/X.25 Gateway.

DECnet/SNA Gateway. A Phase IV DECnet node that connects DECnet–RSX and DECnet–VAX nodes to systems in an IBM SNA network to perform remote job entry, 3270 terminal emulation, or task-to-task communication. There is also software provided for Gateway management tasks. The DECnet/SNA Gateway is implemented in two ways:

- As a node connected to an Ethernet
- As a node connected to a Phase IV DECnet–RSX or DECnet–VAX node in a wide area network.

In both implementations, the functions are the same: remote job entry, 3270 terminal emulation, application programming with an IBM program, and Gateway management. Refer to documentation for each server product described in this section for more detailed information.

General purpose nodes perform a variety of functions that are discussed in Chapters 4 through 9 of this manual. All Phase IV general purpose nodes except PRO/DECnet can run in either a wide area or an Ethernet environment unless otherwise specified. The discussions in this chapter on environments, node routing capabilities, and DECnet phases that do not specifically address server nodes apply to general purpose nodes.

3.3 Line and Circuit Characteristics

Network nodes in a wide area network can be connected using point-to-point or multipoint lines and circuits. Ethernet nodes are connected by means of a bus topology (as noted in Section 3.1.2). The type of line/circuit you have between two nodes can have a marked effect on access capabilities. For example, data on a point-to-point line/circuit between two nodes can travel somewhat faster than data that must be transferred over a multipoint line/circuit.

A point-to-point line connects two nodes using a single circuit. In a point-to-point connection, nodes always have access to the communications path linking them together.

A multipoint line is shared by more than two nodes (see Figure 2–1). Each node communicates over the line using a separate circuit. One node on the line, called the control station, controls access to the shared communications path; the other nodes on the line are known as tributaries.

The way that each node gets to use the communications path in a multipoint connection is through a method called polling. In polling, the control station sends a message to each tributary in an order prescribed by the control station software. The poll message asks if the tributary has any data to send. If the tributary does have data to send, the control station allows the tributary to transmit. When that transmission is completed, the communications path is freed and the control station polls the next tributary. If a tributary has no data to send when it is polled, the control station passes on and polls the next tributary. If data is sent to a tributary from another tributary or system in the network, it must first pass through the control station which will deliver the data (using the routing mechanism) to the specified tributary when the communications path is freed.

A multipoint connection can be more economical than a point-to-point one, as you have several stations communicating over a single line. However, you may not always have access to the communications path when you need it.

Ethernet nodes are linked in a bus topology. This means that Ethernet circuits use a common communications line — the Ethernet cable. The technique by which each user accesses the Ethernet communications path is called Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

Essentially, CSMA/CD states that every communications device associated with a node on the cable must listen for data traffic. If the line is busy, users must wait until it is clear to transmit data. If the line is clear, every user on the cable has equal access rights to it. There is, however, a limit on the amount of data a node can send at one time. This limit prevents any one node from monopolizing the line.

If two or more users simultaneously try to transmit data on the cleared line, a collision of data will occur. Each node senses the collision condition, stops the transmission, and automatically tries to retransmit the data after waiting a randomly selected amount of time. Therefore, all nodes on the Ethernet cable have the same access to it, regardless of where on the cable they are positioned.

3.4 The Total Picture

Figure 3–9 shows an example of a large network that includes a wide area network joined with a local area Ethernet network. As you can see, there are a variety of configurations you can create to serve many needs in such an environment. Nodes in each area of the network can communicate with each other and combine their resources to perform network activities as an active, cohesive unit.

Refer to the software product description (SPD) and installation documentation for your particular DECnet system(s) for more exact configuration guidelines.

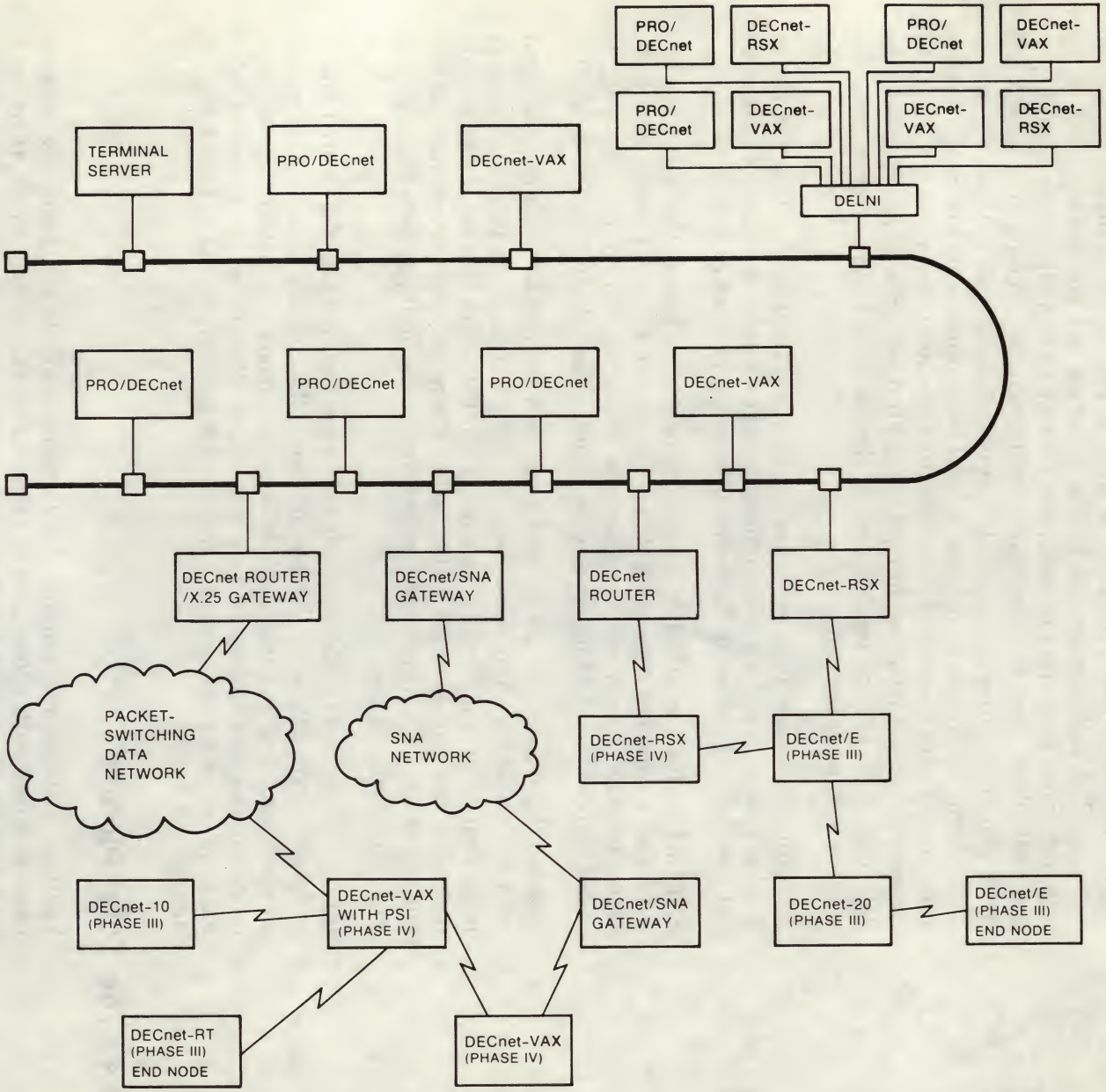


Figure 3–9: The Total Picture

CAPABILITIES AND FUNCTIONS

Chapter 4
Common Mechanisms in DECnet Functions

After configuring your system (Chapter 3) and installing it (Chapter 10), you are ready to perform a variety of network functions. The functions that you can perform with DECnet are divided into the following areas of discussion:

- Task-to-task communication (Chapter 5)
- Remote file and record access and file transfer (Chapter 6)
- Terminal communications (Chapter 7)

Chapter 8 describes how you can perform these functions using foreign vendor networks and Chapter 9 describes applications environments supported by these functions.

All of the functions listed have some common mechanisms at work within them, such as logical links and access control. This chapter discusses those mechanisms.

4.1 Logical Links

Every node runs multiple programs and processes. Processes or programs that want to communicate with each other, whether they are in the same or in different nodes, need some way of establishing contact and exchanging data apart from the confusion and interference of other network traffic. To expedite the orderly flow of information between two programs or processes, DECnet implements a mechanism called a logical link. A logical link is a temporary conversation path established between two communicating programs (or processes) in a DECnet network. (Logical links are not to be confused with lines or circuits; see Section 2.2.)

4.1.1 When Are Logical Links Required?

Almost all network functions, including those discussed in Chapters 5, 6, 7, and 8 require the services of a logical link between programs. Some maintenance functions, such as certain loopback tests and the down-line loading of certain systems, do not (for example, down-line loading an RSX-11S DECnet node, which does not have mass storage facilities). Logical links are used in the following types of connection:

- A user program connected to another user program
- A user program connected to a DECnet module
- A DECnet module connected to another DECnet module

In the first type of connection, the application programmer specifies a series of DECnet calls in each user program to establish and direct the operation of the logical link. In the second type of connection, the programmer includes calls that initiate and control the link in the user program only; the DECnet module automatically handles its end of the link. In the last case, the creation and operation of the logical link is a level removed from the user; user programming is not involved in the connection at all. Typically, two DECnet modules exchange messages based on information supplied by their local system.

4.1.2 How to Create a Logical Link

As mentioned in the previous section, logical links are created by an interaction of DECnet calls specified in the two cooperating programs. This interaction is referred to as the "handshake procedure." In this procedure, DECnet calls are passed between the two programs to establish an agreement to communicate. Unless both programs agree, a logical link cannot be created.

The handshake procedure unfolds in a prescribed order. First, one program, called the source program, issues a call to request communication with a second program, called the target program. DECnet software in the source program's node carries the connection request to DECnet software in the target program's node. (If the two programs are in the same node, only one set of DECnet software is involved in establishing connection—see Section 4.1.3.)

The target program has the option of accepting or rejecting the source program's request for connection. If the target program returns a call accepting the connection request, a logical link is established between the two programs. At this point, source and target distinctions become irrelevant, and both programs can issue calls to send and receive data until either one of them decides to exit or abort the connection. If the target program returns a call rejecting the connection request, no logical link is created.

The individual calls used to establish a logical link and to exchange data with another program are described in the programmer's reference manual for each DECnet system. The programming languages you can use to issue these calls in each system are also listed there. Section 9.1.1 discusses logical link processing in an applications environment.

4.1.3 Behind the Scenes — DECnet Software and Logical Links

The information passed between two programs in the handshaking procedure and in all communications activity is intercepted and evaluated by equivalent DECnet modules in each node. Information passes from the local program to the local DECnet software, from the local DECnet software to the remote DECnet software, and finally to the remote program (as shown in Figure 4–1).

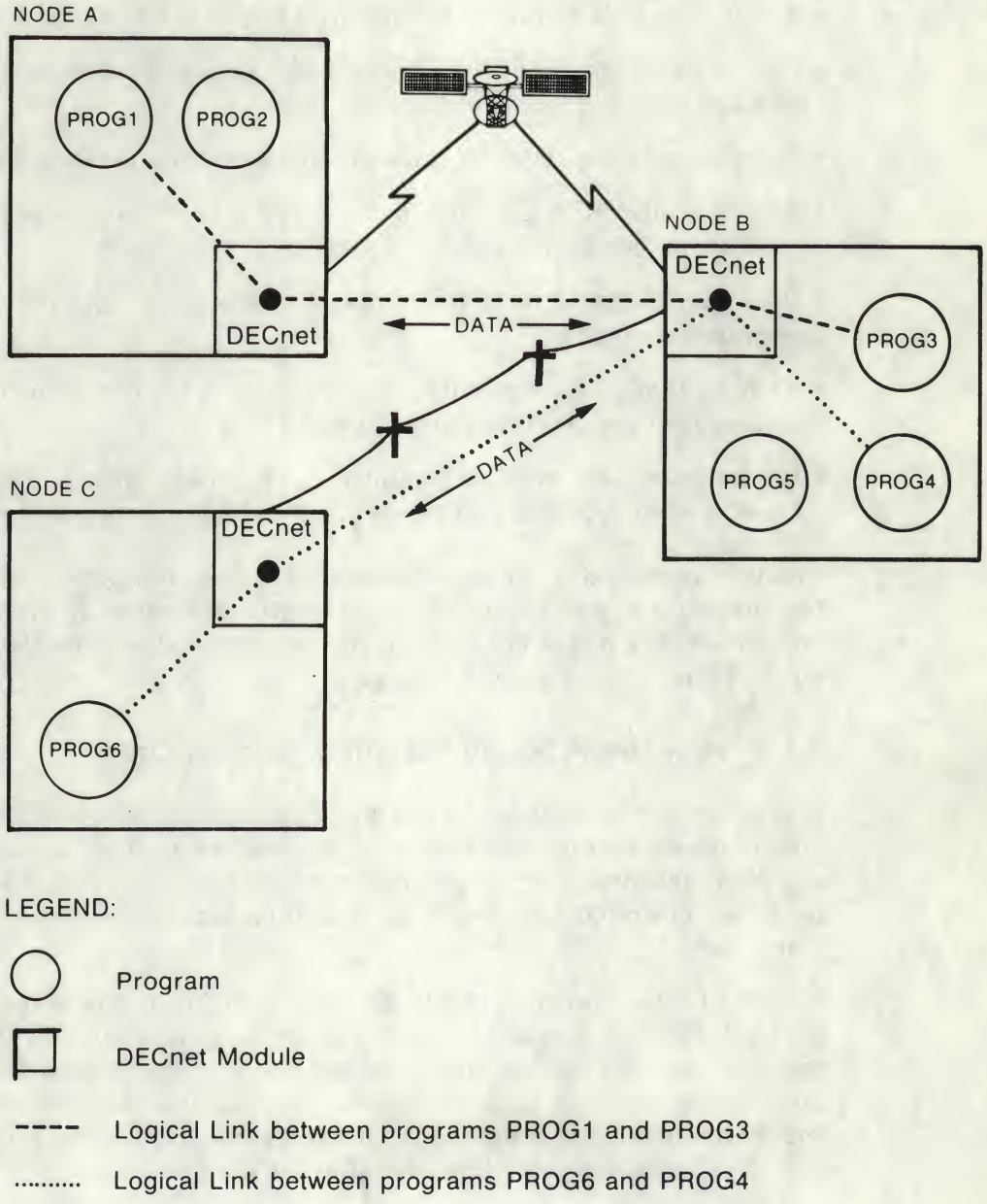


Figure 4–1: Logical Links and DECnet

DECnet software on each node examines calls received from the local program (the program located on the same node) and information received from equivalent DECnet software on the remote node. Each set of DECnet software acts as an intermediary to field invalid information and to pass requests, responses, and data between the two programs in a manner that is transparent to the end user. The user sees only one program talking to the other.

Other actions performed by DECnet software on each node with respect to logical links are:

- Establishes link addresses to identify each end of a logical link
- Checks to see whether the two programs have valid access to each other (see Section 4.5)
- Formats outgoing data for transmission by communication hardware
- Allows multiple logical links to be created over a single physical line by separating data into logical link streams
- Removes incoming data from logical link streams and delivers it to the appropriate program
- Assures that data segments are delivered in proper sequence or are retransmitted (as described in Section 4.3)
- Implements flow control in conjunction with the programs to make sure that there is enough buffer space to receive and store data (see Section 4.4)

The user is not aware of any of these activities; the connection between the two programs appears to be direct and without mediation. However, if some problem occurs in the connection, the DECnet software notifies the user by means of an error or status message.

4.1.4 How the Program Identifies Logical Links

A program can establish and use more than one logical link at a time. The maximum number of links that can run simultaneously in a single program is usually determined by the programmer or by system restrictions. (See your programmer's reference manual for details on the requirements of your particular system.)

In view of this, a program needs some way to differentiate between the multiple logical links that may be running within it. This is handled by having each program specify a unique link identifier during every handshake procedure to identify the link that may be created. The link identifier defines a link only within the context of a particular program; two cooperating programs do not have to specify the same link identifier for the same link.

The DECnet software on each node takes the link identifier specified by its local program, and together, both sets of DECnet software agree on a pair of link addresses that they associate with a particular link. Link addresses are meaningful only to the DECnet software; they are transparent to the end user.

4.1.5 Multiple Logical Links within a Program

A single program can establish logical links that communicate with several programs, or a single program can establish several logical links with the same program to exchange data intended for different purposes. For example, two programs can establish two logical links between them; one link can be used to transmit transaction data, while the other can be used to transmit control information.

4.2 Data Types

Data that you send over a logical link is distinguished as either normal data or interrupt data. Normal data constitutes the subject matter of most data exchanges between two programs. For example, user data transmitted, such as a list of messages, and the responses sent by the receiving program, would constitute normal data flow. To interrupt this flow of normal data, either program can send interrupt data. Interrupt data is typically high priority information signaling the occurrence of some event requiring immediate attention. Interrupt data breaks through the normal data flow. The means of delivering interrupt data are system and program dependent, but the receiver usually reads it before accepting any normal data that may be pending.

4.3 Segment Acknowledgment and Retransmission

The DECnet software that manages each end of a logical link guarantees:

- That all transmitted data is delivered to the destination node in proper sequential order
- That all data received by DECnet software on the destination node is given to the target program in the proper sequence

To guarantee proper segment sequencing, DECnet software numbers the segments it transmits over a link. The receiving DECnet software, using the transmit numbers for identification, must acknowledge the delivery of the segments. If a segment is not acknowledged within a certain period of time, the sending DECnet software retransmits it.

DECnet software assigns a separate set of transmit numbers to interrupt messages. The separate sets of numbers logically divide normal data and interrupt data into separate data streams within the logical link.

The detailed execution of segment acknowledgment and retransmission varies depending on the software implementations involved. However, despite variations in detail, all DECnet software uses these mechanisms to guarantee delivery of all transmitted segments and to ensure that the segments are delivered in the proper sequence.

4.4 Flow Control

Network programs and DECnet software both require a certain amount of buffer space for temporary message storage. For example, DECnet software keeps a copy of every message it sends over a link until the receiver acknowledges receipt of the message. At the program level, buffer space is necessary to hold inbound messages waiting to be processed. DECnet software and programs need buffer space for other purposes as well, depending on the application and the DECnet implementations.

Without some kind of control, message traffic can easily cause available buffer space to overflow. If this happens, communications overhead is incurred because a message must be present at specified intervals until the destination node can accept it. To prevent this, the programs and DECnet software exercise flow control. In most implementations, programs coordinate, send, and receive calls. DECnet software transmits data from a source program only if the target program has issued a receive call. In some implementations, however, the target node's DECnet software must merely have sufficient buffer space available to hold the data. In either case, the DECnet software handling the link between the programs ensures that the appropriate condition is satisfied before any data is actually sent.

The DECnet software modules on each node exchange link service messages to request and convey information about the availability of buffer space and about other conditions that pertain to data flow on the link. The detailed operation of flow control depends on the DECnet nodes on the link. A major aid in ensuring optimum network performance over links and between nodes is a network product called Observer (discussed in Section 11.2).

4.5 Access Control

Most DECnet nodes have some type of access control mechanism that examines logical link requests to prevent unauthorized access of node resources. Typically, petitioning programs must specify information similar to the information needed to log on to the remote system. For example, a program may have to specify an authorized user name or account and password in order to gain entrance to a task on the remote node. DECnet software at the remote node verifies the information given and decides whether or not to allow access. In task-to-task communication, you must specify access control information in a special parameter when you request a logical link connection (see Section 5.2). In remote file access (Section 6.1.4) and in terminal communications (Section 6.2.1), you must include this information in the file specification. (File specifications differ for each type of DECnet system; therefore, refer to the programmer's reference manual for your DECnet system to determine correct syntax.)

Access control information may not be required for every network function. To determine what information is needed and when, consult the system manager at the remote node to which you are seeking access. Access control information is initially established during installation, and can be dynamically modified on most systems by the system manager.

Chapter 5
Task-to-task Communication

DECnet enables two programs within a network to perform task-to-task communication — that is, to exchange data over a logical link. For example, an RSX–11M program can use local DECnet–RSX facilities to communicate with a program running in a DECnet–VAX node. (There is no DECnet user interface for task-to-task communication using the Professional 300 series personal computer. Therefore, the rest of this discussion may not be relevant for PRO/DECnet users.)

In most DECnet implementations, performing task-to-task communication is similar to performing input/output (I/O). The logical link between the two programs is like an I/O channel over which both programs can send and receive data. (Refer to system-specific documentation for details on how a particular operating system implements I/O operations.)

5.1 DECnet Task-to-task Calls

DECnet calls are written into cooperating task-to-task programs to enable them to communicate with each other. These calls activate routines requesting the local DECnet software in each node to perform specific functions such as the creation and control of a logical link (see Chapter 4).

The form of the DECnet calls that a programmer specifies depends on the programming language being used; calls can actually be calls, macros, or system directives. However, the DECnet task-to-task capability translates each of these call types into the same set of DECnet software messages regardless of the language in which they are programmed. For example, the DECnet software sends the same type of message to the remote node in response to a FORTRAN connect request call as it does in response to a connect request macro. Therefore, you do not have to be concerned about what programming languages are supported by the remote node when you write a task-to-task program.

You do, however, have to be concerned about what languages are supported by the local node since not all operating systems support the same languages and all languages do not support task-to-task communication. Refer to the programmer's reference manual for your particular DECnet system for a list of the programming languages you can use for task-to-task communication.

Every DECnet task-to-task program can issue calls to perform the following functions:

- Request a logical link
- Receive a logical link request
- Accept or reject a logical link request
- Send data
- Receive data
- Send interrupt data
- Receive interrupt data
- Terminate the logical link

Depending on the local system and the language used, there is not always a one-to-one correspondence between the call issued and one of the above steps. In some cases, a program must issue three separate calls to create a logical link; in other cases, a program needs to issue only one call.

5.2 Requesting a Logical Link

The first thing you need to do to establish task-to-task communication is to request a logical link. To do this, you issue a connect request call according to the language conventions of your particular DECnet system. (This is the first step in the handshake procedure described in Section 4.1.2.)

The connect request call must provide network addressing information that identifies who is sending the message and who should receive it. You can specify this addressing information in the source program either by building a special data area (called a connect block) or by specifying an ASCII string called a network specification.

5.2.1 Building a Connect Block

Most DECnet task-to-task call sequences include one or more calls to build a connect block. You specify parameters to the connect block call(s) that identify the source and target programs. These parameters will include all or part of the following information:

- **Link identifier.** This identifier differentiates the requested link from any other links currently being used by the source program (see Section 4.1.4 for a discussion of logical link identifiers). If the connect request succeeds, the source program uses the link identifier to address data to be sent over the link. The source program's link identifier is called by various names, depending on the operating system implementing DECnet. Check the programmer's reference manual for your system to determine what the link identifier is called on your system.
- **Target node identifier.** This identifier can be a unique node address that distinguishes the node from all others in the network, or it can be a locally defined node name that the local DECnet software translates into a unique node address (see Section 2.1).
- **Object type or name.** Object is another term for a network program. A network program or object has a special identifier for use in network calls. This identifier consists of an object type and/or an object name. Section 5.2.1.1 explains the significance of the object type and name. Figure 5–1 shows how objects are addressed.
- **Access control information.** This information describes the source program and includes a user identification, a password, and optionally, an account number. The information is equivalent to the data a user supplies when logging on to a system. In most implementations, the target program uses this information as a factor in its decision to accept or reject the connect request.
- **Optional user data.** A source program usually has the option of sending 16 bytes of data to the target program as part of a logical link connect request.

When you finish building the connect block using the call(s), macro(s), or directive(s) appropriate for your system and language, you can specify the label or address of that connect block in the source program as a parameter to a connect request call. This provides network addressing information you need when requesting a logical link.

5.2.1.1 Object Types and Names — As mentioned in the previous section, one of the parameters you specify in a connect block is an object type or name. An object type or name identifies — to the local DECnet software — a program that provides a known service and uses known protocols. For example, object 17 (FAL) provides remote file access.

In most DECnet implementations, a program must declare its object type and name to the DECnet software in order to be eligible to receive link requests. In some implementations, a system manager can use a DECnet command at a terminal to declare a program's object type and name. (Check your programmer's reference manual to find out how objects are specified on your system.)

The object name may be a special network name for the program, or it may be the same name by which the program is known to the local operating system. Object types and names are used by the source program in a connect request to specify the target program with which it wants to communicate. This is done using one of two formats:

- An object type equal to 0 and an ASCII name
- An object type equal to a positive integer (from 1 to 255) and a null name

The first format identifies a program by name, whereas the second format identifies a program by numeric object type.

To address a target program, a connect request specifies either a name or an object type, but not both. The first format — object type 0 plus name — is commonly used to address user-written network programs. To use this format in a connect request, a source program must know the target program's declared object name. Note that the maximum length allowed for a name depends on the local node's operating system.

The second format provides an abbreviated means of identifying a frequently used network function, usually a specific DECnet module. A specific type always represents the same generic function within a network, even if the program that actually performs the function has a different name at each node. In this way, a network program can address a function without knowing the function's name in the target node.

Digital reserves a range of object types for DECnet system programs. Types within this range are used consistently across all DECnet implementations to refer to the same functions. The programmer's reference manual for each DECnet implementation lists the object types reserved for DECnet use.

Unreserved types can be used for user-written network programs. For example, in a user-written transaction-processing application, each node might have a resident program for recording statistics on transactions performed within the last 24 hours. The application's designer could choose an unreserved object type to identify all such programs throughout the network.

Figure 5–1 illustrates the object identifiers for several programs in Node A and Node B.

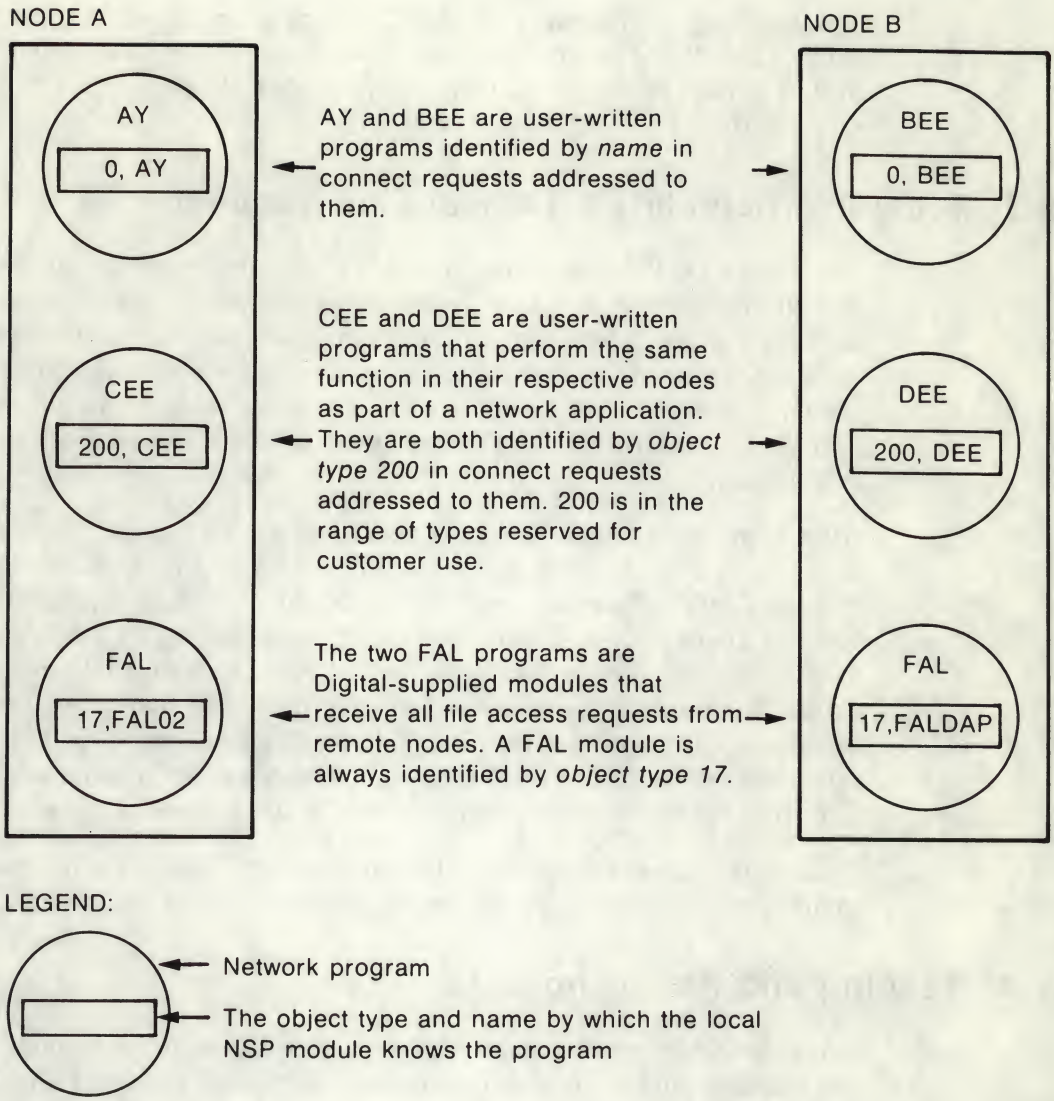


Figure 5–1: Addressing Network Objects

5.2.1.2 Access Control — Access control is implemented in different ways on different systems. Refer to documentation for your system for specific information. A general discussion of access control is provided in Section 4.5.

5.2.2 Network Specifications

If you do not provide network addressing information for the connect request using a connect block, you must provide that information in a network specification. (Some DECnet systems require a network specification instead of a connect block. Refer to documentation specific for your DECnet system to find out if and how network specifications must be used.)

A network specification is an ASCII string specified in the connect request itself. Like the connect block, it includes a target node identifier, access control information, and a target object type or name. (DECnet–20 network specifications can include optional data.)

5.3 Accepting/Rejecting a Logical Link Request

The source DECnet software forwards the connect request to the node specified in the connect block or network specification. The DECnet software in the target node checks that the program addressed in the connect request is a valid object and then verifies the source program's identification (see Section 5.2.1). If the target program is a known object and any required verification checks out, the target DECnet software delivers the connect request to the target program.

After examining the incoming connect request, the target program either accepts or rejects it. (Transparent communication in DECnet–VAX and the concise COBOL interface in DECnet/E do not allow a program to reject a connect request. See system-specific documentation for details on how each system handles connection requests.) The target node's DECnet software forwards the appropriate response back to the source node. The target program usually has the option of sending 16 bytes of data along with the acceptance or rejection of the link. For example, a connect reject response might include data that tells the source program why the connect request was not accepted.

If the target program agrees to the connection, it specifies its own logical link identifier in a call accepting the request to create a logical link.

5.4 Sending and Receiving Data

Once the target node accepts the connect request and a logical link has been established, either program can send and receive data using the assigned logical link identifiers as address parameters on each message. They can alternately send and receive, or they can send and receive simultaneously.

Two kinds of data can be sent over a logical link: normal data and interrupt data. As Section 4.2 explains, normal data makes up the subject matter of the programs' dialog, whereas interrupt data conveys special high priority information.

5.4.1 Normal Data

To convey normal data over the link, a source program issues one or more calls to send the data, and a target program issues one or more calls to receive it. The DECnet software at the source node will not transmit data unless the target program has already issued a receive call. Each receive call allocates the buffer space needed by the target program to store the data. Figure 5–2 outlines this procedure.

Some DECnet implementations allow the source DECnet software to transmit data over any logical link as long as the target DECnet software has access to enough system buffer space to hold the data. The DECnet software at the target node then delivers the data it has received when the appropriate target program allocates its own buffer space by issuing one or more receive calls.

Regardless of the implementations involved, the DECnet software in the source and target nodes exchange link service messages to determine whether the target program is prepared to receive a message. This precaution is part of DECnet's flow control mechanism (described in Section 4.4).

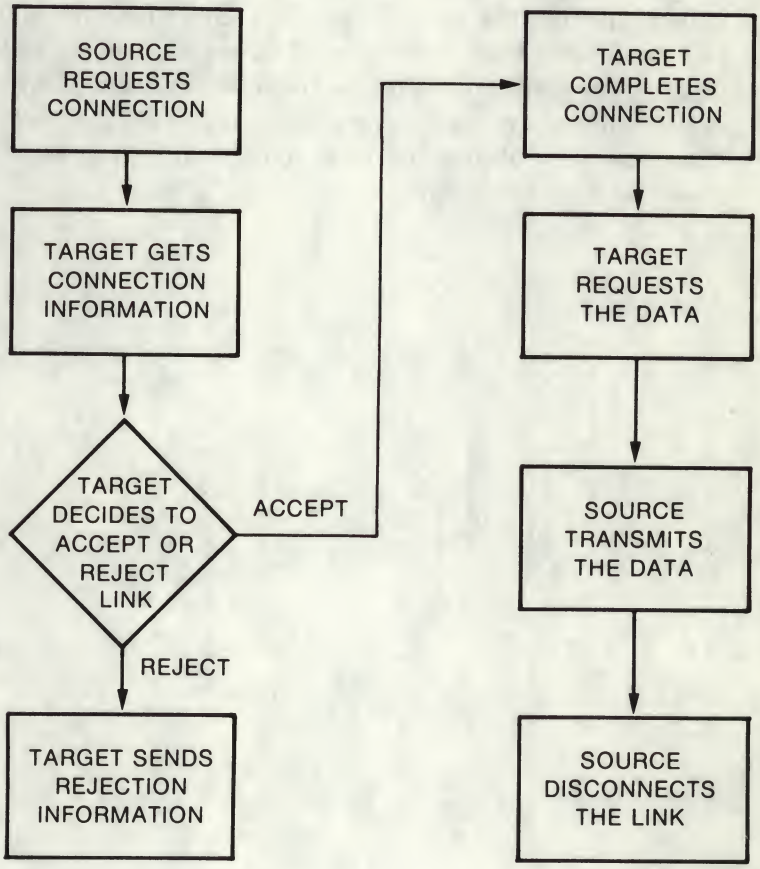


Figure 5–2: Transmitting Normal Data

5.4.2 Interrupt Data

Interrupt data can consist of up to 16 bytes of information to be delivered immediately to the target program. If the target DECnet software has a queue of normal data already received but not yet processed, the interrupt data is placed either at the head of that queue or in a separate queue that the target program can access without first reading the normal data.

5.5 Terminating the Link

Either program can issue a call at any time to terminate the link in one of two ways. One way, which disconnects the link in an orderly fashion, is normally used by a program to terminate a session that has proceeded as expected. All pending transmissions are completed before the link is dissolved. The programmer must decide which of the two programs disconnects under normal circumstances. When discussing these orderly disconnections, the various DECnet user's manuals call them disconnects or synchronous disconnects.

The second way to disconnect forces the link to be terminated whether or not the remote DECnet software has acknowledged previously transmitted data. In most programmer's manuals, this method of disconnecting the link is called aborting the link. When the caller's DECnet software receives notification to abort a link, it cancels all messages waiting to be transmitted over the link. A program may choose to abort in response to some unusual system event, such as an impending emergency shutdown. Whether a program simply disconnects or aborts the link, it can simultaneously send up to 16 bytes of data to the other program.



Chapter 6
Remote File and Record Access

Using DECnet, a program in one node can access a file in another node, despite differences in the operating and file systems of the two nodes. This remote file access capability has the following applications:

- A user-written program can incorporate DECnet I/O calls that allow it to perform the following record-level operations: (1) open and close a remote file (2) create or delete a remote file (3) read from or write records to a remote file. These operations are described in Section 6.1.
- A terminal user can run a DECnet utility or issue a system command to manipulate remote files as follows: (1) transfer (copy) files to or from remote nodes (2) delete remote files (3) submit and execute command files at a remote node (4) append files to local or remote files (5) list remote directories (6) queue files to a remote printer. These operations are described in Section 6.2.

Like other DECnet functions, remote file access requires the cooperation of two network programs, which exchange a series of DECnet messages. A program in one node issues a call requesting remote file access. This call is translated into one or more DECnet messages by the local DECnet software and then sent to a program in the remote node. The program in the remote node receives the file access request messages and translates them into a form recognizable to its file system. If the request is accepted, the file operations are performed.

As in task-to-task communication, the two programs must establish a logical link before they can begin communicating. See Section 4.1 for information on logical links.

In the context of remote file access, the program that requests remote file access is called the source program, and a DECnet system program that receives the request is called the target program. Depending on the type of DECnet system you are using, the source program can be one of the following types of software modules (refer to system-specific documentation):

- A version of the Network File Transfer (NFT) utility
- A system command such as the COPY command
- A user-written program that accesses remote files via calls to DECnet subroutines or to file access subroutines provided by the local file system

The target program is a DECnet utility (generally referred to as a file access listener) that translates file access requests it receives into calls to its local file system. The target program is also responsible for sending any resulting file data back to the source program. File data sent by the target program is first received by DECnet software which presents it to the source program in a format compatible with its local file system.

File data can flow in either direction between the source program and a target program. The file accessed can reside on a mass storage device like a disk and it can be directed to an I/O device such as a line printer or a terminal. The actual file operations that a source program can perform depend on the file systems local to both nodes. See the programmer's reference manual associated with the particular systems involved in remote file access for detailed information.

Section 6.1 discusses how a program can issue calls to gain access to remote files. Section 6.2 describes how to access remote files from a terminal. See Section 9.2 and 9.3 for examples of DECnet's remote file access capability in an applications environment.

6.1 Programming Remote File Access

To gain access to a remote file, a user program incorporates DECnet I/O calls, which activate remote file access subroutines. The function of these subroutines is to build, send, and interpret DECnet file access messages. All DECnet systems allow you to perform the following operations on a remote file:

- Open an existing file
- Create a new file
- Read records from a file
- Write records to a file
- Close a file
- Delete a file

In addition to these operations, specific DECnet systems allow user programs to manipulate remote files in other ways. The programmer's reference manual for each system discusses what operations are available and how you specify DECnet calls for remote file access on a specific system. Sections that follow discuss factors in preparing to write remote file access programs that apply to all DECnet implementations.

6.1.1 File System Capabilities

A programmer needs to be familiar with the file system resident at the target node. File organization, access modes, and other characteristics are dependent on the type of file system that each operating system supports. When the local and remote nodes have the same operating and file systems, programming remote access is similar to programming local I/O operations. However, when the access operation bridges different types of operating systems, the programmer faces certain variables and restrictions.

Generally, the theory of the lowest common denominator applies: a source program can perform those functions that its local language provides and that the remote file system supports. For example, a VAX/VMS program cannot use all the remote access functions provided by DECnet–VAX if the target node runs RSX–11M.

By cross-checking the file system characteristics with the available remote access calls, a programmer can find out the kinds of file operations that are possible between two different DECnet implementations. This information is provided in the programmer's reference manual for each DECnet system.

6.1.2 Initiating Remote Access

Like task-to-task communication, remote file access requires a handshake sequence at the beginning of the operation. Not only does DECnet software set up a logical link between the source and target programs, it also exchanges DECnet file access messages to prepare for the file operation that will be performed over the link.

This extended handshake, which is transparent to the source program, happens automatically when the program issues a call to open a remote file. The form of the open call varies from system to system and from language to language, but the call always provides much of the information exchanged in the handshake. Using the information supplied by the call, as well as system-supplied data about the local file system, the remote access subroutines generate DECnet file access messages addressed to the target program. In response, the target program sends back DECnet file access messages to define characteristics of its local file system.

For most remote access operations, the open call issued by the source program passes the following information to subroutines within the local file system:

- A file specification
- Access control information
- Characteristics of the file to be accessed

Depending on the open call's particular function — for example, open for reading or open for appending — the call passes other information as well.

6.1.3 The File Specification

The file specification identifies the remote file to be accessed. Because the remote file system actually carries out the requested file operation, the programmer must know how the file is identified by users on the node where the file resides. Refer to the programmer's reference manual for the source and target systems for file specification syntax.

6.1.4 Access Control Information

Access control information identifies the program to the remote system and consists of:

- A user identification code or name
- A password associated with the user identification
- Additional accounting information as required by the remote system

If this information matches an account or guest account entry in the remote system's user file, the program gains access to that system's resources. Like the file specification, the access control information must be recognizable to the remote system and therefore specified according to its syntax.

Gaining access to the remote system does not guarantee that requested file operations will succeed. In most Digital operating systems, each file has a corresponding protection code that determines the types of access allowed to defined groups of users. The user identification — a code or a name — specified by the accessing program determines the program's group category and therefore determines the types of access it can make to each file.

6.1.5 File Characteristics

File characteristics define the file to be accessed in the following ways:

- **Access method — sequential or random.** The program requesting file access indicates how it will access the file. All Digital file systems support sequential access in which each I/O operation reads or writes the next record. Selected file systems permit random access, which allows the program to access a specific record anywhere in the file.
- **File organization — sequential, relative, or indexed.** Sequential file organization means that records in a file are arranged one after the other. This organization is supported for all types of devices. Relative file organization means that records within a file are identified by a relative record number. This number identifies the record's position relative to the beginning of the file. Indexed organization (RMS only) is a complex file structure that allows both sequential and random access and uses record keys for identification. The keys used to identify individual records are defined at file creation. Relative organization is supported only for disk devices.
- **Data type — ASCII or image.** ASCII data is subject to formatting conversion by the DECnet software, depending on the data's record attributes (see below). Image data is a stream of bits to which the DECnet software applies no interpretation.
- **Record attributes.** This characteristic indicates the type of vertical format control that applies to the file.
- **Record format — fixed length, stream, variable length, or variable with fixed length control (VFC).** This characteristic indicates the way the records are formatted within the file. A VFC record, supported by RMS only, includes a fixed length control field in addition to the variable length data portion.

6.2 Accessing Remote Files from a Terminal

All DECnet implementations support terminal-based access of some kind to remote files. Some DECnet implementations use system commands such as COPY, TYPE, APPEND and so on. Other DECnet implementations use the Network File Transfer (NFT) utility. Refer to your programmer's reference manual to determine which access method your system supports for terminal-based file access.

Regardless of which method is implemented on your system, as a terminal user, you can perform the following file operations:

- Transfer (copy) a file to or from a remote node or between two remote nodes.
- Delete a remote file.
- Submit a local command file for execution at a remote node or execute a command file already existing at a remote node. (Command files cannot be submitted to or executed at a DECnet-RT node.)
- Append one or more local or remote files to an existing local or remote file (on DECnet-20 nodes, supported only for RJE-20 stations).
- Obtain listings of remote directories. A directory file lists all the files residing on a device that belongs to a specific user or category.
- Queue one or more files to a line printer. The files can be remote and the printer local, or the files can be local and the printer remote (on DECnet-20 nodes, supported only for RJE-20 stations).

(Some systems supporting system commands rather than NFT also enable you to create, open, and close remote files as well as read to and write from them. Check system-specific documentation for details.)

On behalf of the terminal user, the access method (be it NFT or system commands) creates a logical link between itself and the remote DECnet software. From the input a user types at the terminal, the access method formulates the appropriate network messages, which it then sends over the link. In turn, the target program interprets the messages it receives and interfaces with its local file system as requested. The target program then returns information and any requested file data to the source program by sending back network messages. See Section 9.4.2 for examples of how this works in an applications environment.

6.2.1 Access Control

A terminal user must supply the same access control information that a program must pass to perform remote file access. This information — user identification, password, and optional account data — must be available whenever a file access command refers to a remote node. Refer to system-specific documentation to see how access information is handled in terminal-based file access on each system.

6.2.2 File Protection

The access control information supplied locally by the terminal user determines the user's access rights at the remote node. As Section 6.1.4 describes, the remote file system compares the user's identification with the protection code associated with the file to be accessed. The file system carries out a requested access only if the protection code for the file grants access to the identified user.

6.2.3 Remote File Specifications

A specification that describes a file on a remote node must conform to the remote node's syntax rules. If the remote node has a different operating system, the remote file specification may contain fields or conventions that the local system does not recognize.

To prevent the local node from being confused by a foreign syntax, some implementations require the user to enclose the foreign specification in double quotes; some do not. Refer to the programmer's reference manual for your system for specific guidelines.

6.2.4 Remote Command File Submission

Most terminal-based file access implementations allow a terminal user to execute a command file in a remote system. A command file contains ASCII command lines equivalent to the command lines a user enters at a terminal. The remote node reads and executes these commands when a user submits the file. See system-specific documentation for details on how this is performed by systems that support the option.

Chapter 7
DECnet Terminal Facilities

DECnet implements several terminal facilities for interactive access to the network. These include:

- Interactive terminal-to-terminal communication
- Direct access to a remote node's operating system using a network command terminal facility
- Remote file access from a terminal

This chapter presents overviews of terminal-to-terminal communication and direct access to a remote node's operating system. Accessing remote files from a terminal is discussed in Chapter 6.

7.1 Interactive Terminal-to-terminal Communications

Interactive communication between two DECnet terminals can be accomplished in several ways:

- **Phone.** The VAX/VMS operating system has a utility called Phone that can be implemented by DECnet to allow a terminal-to-terminal conversation over the network between two DECnet–VAX terminal users.

 Phone allows you, as a terminal user, to "dial" another terminal user on the same or on a different DECnet–VAX node. In a DIAL command, you specify the name of the target node and the name of the person you want to talk to. If that person is logged on and available, the Phone utility notifies them of the pending phone call by activating the bell function on the remote terminal and by displaying messages on that person's terminal screen. The person you are calling can then elect to answer your call by issuing an ANSWER command. Once your call is answered, the two of you can type messages interactively from your terminals in the manner of a typed conversation. At the end of the conversation, either of you can issue a command to hang up.

Phone has a number of options that can be used to provide other services and information to terminal users. For example, there is a directory service listing available users on each DECnet–VAX system supporting Phone.

NOTE

DECnet terminal users involved in communications using the VAX/VMS Phone utility must be sure that the node they are trying to communicate with also supports it. Nodes supporting different, although similar, utilities cannot communicate at this level.

- **TLK.** Some DECnet systems support a DECnet utility called TLK. TLK is installed as part of DECnet. It enables terminal users on two DECnet–RSX systems to communicate. Using TLK, you can send single messages to another terminal user (single line mode) or you can engage in an interactive conversation (dialog mode). To send a message in either mode, you issue a TLK command specifying a node name and terminal identifier for the terminal you wish to "talk" to. The user at the other terminal receives a copy of the message(s) sent, along with the identity of the sending terminal. All messages received are displayed on the terminal screen.

 Single line mode messages contain just one line and are terminated when you type a carriage return to send the message. To end a "conversation" of dialog mode messages, either user can type a CTRL/Z or EXIT command.

 You can also use TLK to send messages read from a command file. TLK command files are useful for sending many messages at once and for storing and sending sets of messages that need to be sent more than once.

 TLK is not supported on all DECnet systems. Refer to user documentation for your system to see if TLK is supported, and if it is, for details on how to use it. Refer to Section 9.4.1 for an example of how interactive terminal communications can be used in an applications environment.

7.2 Network Command Terminal Facilities

All DECnet systems, with the exception of DECnet–RT, have a utility that logically connects a local terminal to a remote node operating system of the same type. If you have a Phase III DECnet system that supports a network command terminal facility, you can communicate with other DECnet nodes of the same type. For example, a DECnet–20 terminal user could use the SETHOST terminal facility to log on to another DECnet–20 node; a DECnet/E terminal user could communicate with a remote DECnet/E node using the utility called Network Command Terminal (NET). (Intermediate routing nodes can be any DECnet implementation.) Refer to system-specific documentation for details on how to use available network command terminal utilities.

Although different DECnet systems can support different network command terminal utilities, each one allows a terminal user issuing appropriate commands to temporarily become a "local" user of a specific remote node. The network command terminal utility or command in the source node sets up a logical link with a cooperating program in the remote node. The resulting connection allows the user to perform most functions that the remote node allows its local users to perform (see Figure 7–1 for an example).

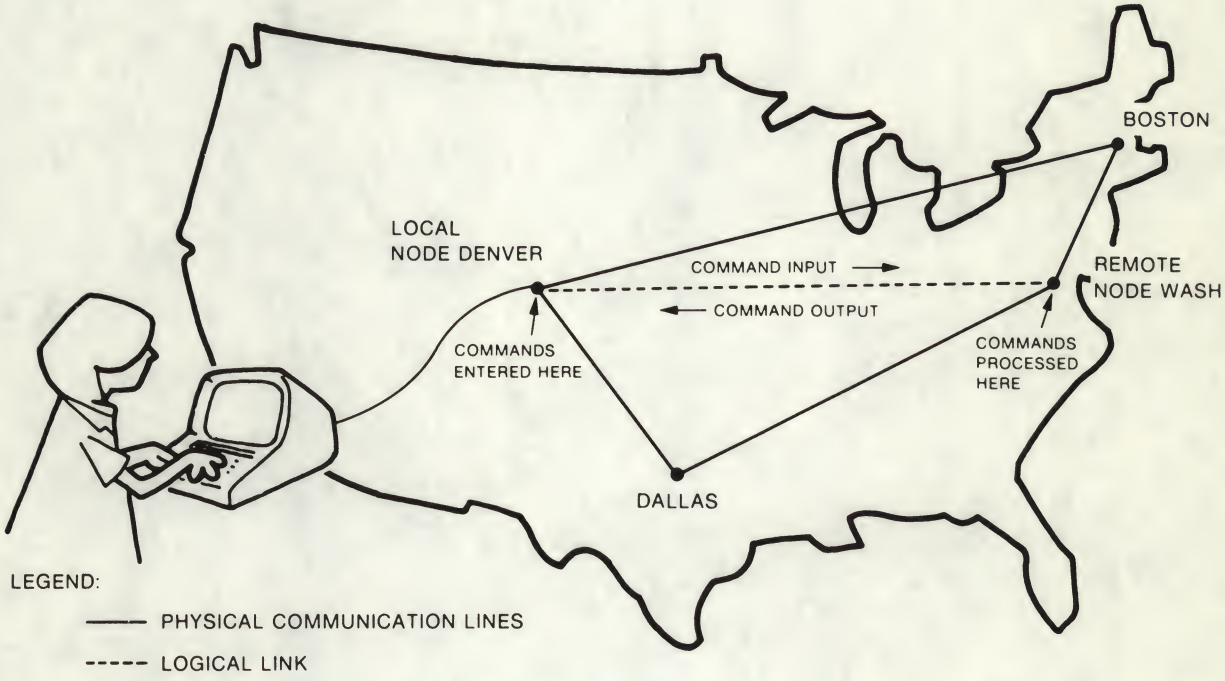


Figure 7–1: Remote Terminal Processing

The ability to set up a logical connection between a terminal and a remote node has numerous applications. Remote, interactive program development is possibly the most common and important application. If the local node does not have the resources necessary to support program development (a DECnet–11S node, for example, has no mass storage facilities), a programmer can obtain access to a remote node that has the required resources.

Phase IV DECnet provides an enhanced network command terminal function referred to as the network virtual terminal (NVT) service. As in Phase III DECnet, NVT terminal users can connect to nodes having the same kind of DECnet system. However, NVT provides the enhanced capability of allowing Phase IV DECnet terminal users, either on a DECnet node or on a terminal server, to connect to other Phase IV nodes of different types that support NVT. Refer to documentation for your system to ascertain whether or not it supports NVT capabilities.

Chapter 8
Internetwork Communications

DECnet enables you to perform internetwork communications — communications with a network other than DECnet. Specifically, you can communicate with DECnet and non-DECnet systems over an X.25 packet-switching data network (PSDN) and with systems in an IBM SNA network.

In order to communicate with DECnet or non-DECnet systems over a PSDN using X.25 communications, you need either a DECnet system with Packetnet System Interface (PSI) software installed or a DECnet system that has an X.25/X.29 extension package (XEP) installed and can be connected by DECnet to a Router/X.25 Gateway (see Section 8.1.3).

To communicate with an IBM mainframe in an SNA environment, you need a DECnet/SNA Gateway node and DECnet/SNA software subroutines installed in your DECnet node (see Section 8.2).

This chapter describes briefly what these additional software components are and how you use them in internetwork communications. For further details, refer to system-specific documentation.

8.1 X.25 Communications

X.25 is a recommendation of the Comite Consultatif International Telephonique et Telegraphique (CCITT) which defines a standard means for computers to interface with packet-switching data networks. A packet-switching data network (PSDN) is a data communications service offered by common carriers, such as the Postal Telephone and Telegraph Authorities (PTT). Packet-switching data networks are widespread, with the number increasing every year. They are a viable means of sharing telecommunications costs and services with users outside of your own network.

NOTE

While all public packet-switching data networks must use the X.25 interface, private carriers can also run a packet-switching network service using X.25 standards.

As the name indicates, a packet-switching data network is one that receives addressed packets of data from network users and conveys them through internal switching routines to specified receivers. The way the network delivers the packets is completely transparent to the end user who has no influence over the path the packet takes through the network. The way that the data packets get from the user to the PSDN is what the X.25 recommendation outlines.

The X.25 recommendation defines standards that govern the relationship between users and the PSDN network on the following levels:

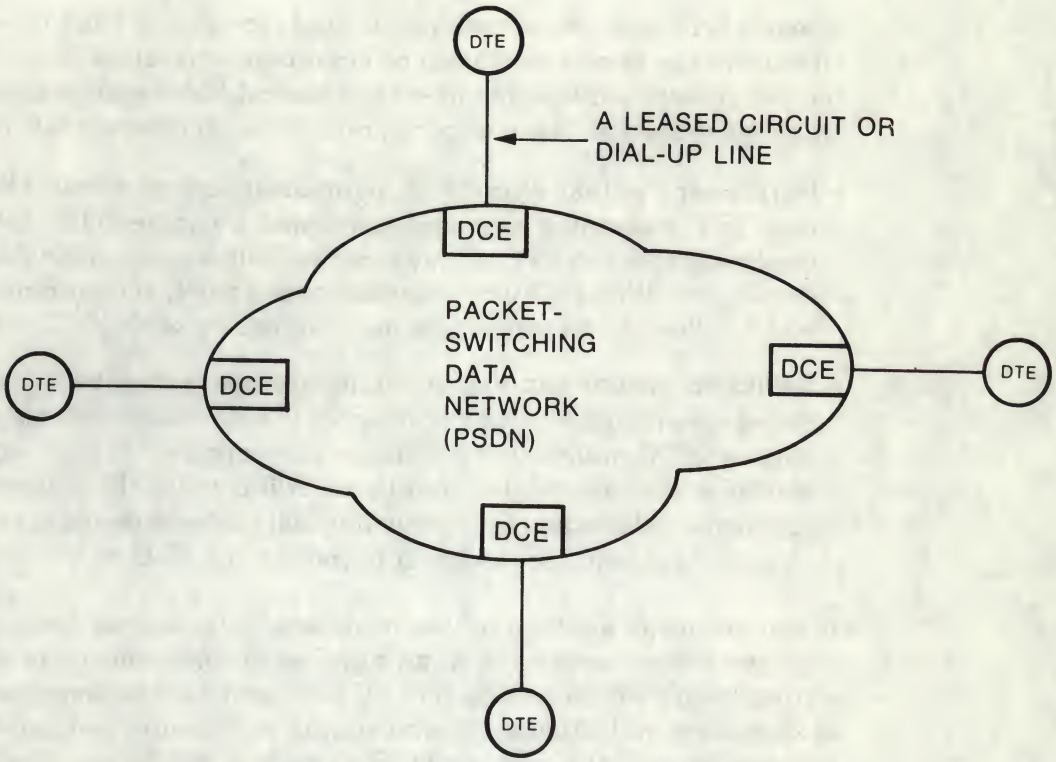
- **Level 1, the physical level.** Defines the mechanical, electrical, functional, and procedural characteristics of the physical link between the network user's equipment and the PSDN network equipment
- **Level 2, the frame level.** Defines the link access procedure for a data exchange over the communications link between the user and the PSDN
- **Level 3, the packet level.** Defines the packet format and procedures for the exchange of packets containing control information and user data between the user and the PSDN

Following the X.25 recommendation, a header containing control and destination information is included as part of each data packet submitted to the PSDN network. (In DECnet implementations, X.25 software described in Section 8.1.3 creates this header.) The header information tells the PSDN who is sending the data and who is to receive it. That way, no matter what kind of system or language is used to generate the data message, the PSDN can recognize the header information and route the packet to its proper destination.

The PSDN shares its transmission lines by interleaving packets from many senders. This provides fast and economical service. The network differentiates between multiple packets on a line by means of a numbering scheme that assures the packets get sent to the proper destination in the correct order.

8.1.1 DTEs and DCEs

Each PSDN consists of a number of geographically separated switching nodes that are connected by high speed links. When you lease a circuit from the PTT or other carrier, the circuit connects your computer to one of the PSDN switching nodes. The PSDN switching nodes are called network interfaces or data-circuit-terminating equipment (DCE). User computers or terminals connected to DCEs are called data terminal equipment (DTE). Figure 8–1 illustrates these components.



LEGEND:

DCE = Data circuit-terminating equipment, a PSDN switching node

DTE = Data terminal equipment, a user computer or a terminal using the PSDN

Figure 8–1: Components of a Packet-switching Data Network

8.1.2 X.25 Circuits

Two DTEs (for example, your system and the one you want to communicate with over the PSDN) communicate by means of one or more virtual circuits. A virtual circuit is a logical association set up by the PSDN either permanently or temporarily. Each virtual circuit handles the exchange of data between two specific DTEs. The DTE at each end assigns a logical channel and a corresponding channel number to the circuit. When sending data, the DTE includes a logical channel number to identify the channel and the corresponding circuit to which the data belongs.

When a DTE uses more than one virtual circuit at a time (that is, multiple circuits to the same destination or circuits to several different destinations), the circuits are multiplexed over the physical link between the DTE and the DCE. Virtual circuits are either permanent or temporary (switched):

- **Permanent virtual circuits.** A permanent virtual circuit (PVC) is analogous to a leased line between a local and a remote DTE. Either DTE can send data over the PVC at any time without issuing calls to set up or break the circuit. When a user subscribes to a PSDN, the administrators of the PSDN allocate the logical channel number for each PVC to be used.
- **Switched virtual circuits.** A temporary association between two DTEs is called a switched virtual circuit (SVC). A DTE sets up this type of circuit only when it wants to send data. The sending DTE assigns a channel, identifies the target DTE, and then obtains that DTE's agreement to communicate. When the DTEs have finished exchanging data, one or the other initiates a clearing sequence to terminate the SVC.

If you are using an X.25 native mode user interface as discussed in Section 8.1.3, you must issue calls in an application program to specify or create a virtual circuit with a remote DTE. If you use data link mapping (DLM) mode as discussed in Section 8.1.3, you simply include the node and task name of the destination DTE and the DECnet routing mechanism handles the selection of circuits as it would over any DECnet-to-DECnet connection. (Using DLM, the user is not even aware that the connection with the destination node is occurring over a PSDN.)

8.1.3 X.25 Access Methods

Access methods refer to how you establish communications over a PSDN. As mentioned previously, you need specific software and, in some cases, hardware products in your network for X.25 communications. What you need depends on with whom you want to communicate. You can access a PSDN to communicate with other DECnet systems or with non-DECnet systems.

If you communicate only with DECnet systems, you may be able to use a DECnet facility called data link mapping (DLM). (Check system-specific documentation to see if your system supports DLM. Some systems, such as DECnet–10 and DECnet–20 do not.)

DLM allows two DECnet nodes to communicate as though the PSDN were not even there. In fact, the X.25 interface is completely transparent to both users. To use DLM, you must be able to send your data through a DECnet node that has PSI software installed. This node must be positioned as a DTE in relation to the PSDN. If you do not have access to such a node, you can send the data through a DECnet Router/X.25 Gateway node on an Ethernet. Using the DLM interface, your node can be located anywhere on the network and does not require any special software other than DECnet.

If you want the option of talking to DECnet or non-DECnet systems over the PSDN, you need different software. RSX-11, VAX/VMS, and TOPS-20 operating systems have PSI software that you can install into your corresponding DECnet system. This software allows you to communicate in X.25 native mode if you are directly connected to a PSDN. (TOPS-20 software is divided into two categories: [1] PSI gateway software installed in the DN20 front-end processor, which provides gateway functions [2] PSI access routines installed in the DECsystem-20 main processor. The access routines communicate with the gateway software to set up a connection with the PSDN. Because the DN20 is the DTE in these connections, it must be directly connected to the PSDN.)

X.25 native mode essentially provides the complete X.25 programming interface, which can be interpreted by any system on a PSDN regardless of vendor. As long as the remote DTE uses the X.25 protocol, which it must as a requirement of the PSDN, then you can communicate with it.

Using PSI software in native mode enables you to write application programs to access resources on the other DTE. This involves issuing calls to create or specify virtual circuits, send and receive data, and terminate communications. Refer to PSI documentation for specific operating systems for a list of supported calls, functions, and languages you can use.

PSI software in DECnet-VAX nodes and DECnet-RSX nodes also provides the DLM interface for communication with DECnet systems over the PSDN. DECnet nodes containing PSI software do not require the services of intervening DTEs or DECnet Router/X.25 Gateway nodes.

If you are using a DECnet-VAX node and do not want to install PSI software, you can install a VAX-11 X.25/X.29 Extension Package (XEP) into your node. The extension package provides subroutines that enable you to use DECnet communication facilities to create and transmit data packets containing X.25 information. The XEP data packets that you send are received by corresponding X.25 software in a DECnet Router/X.25 Gateway; you cannot use the XEP without the services of the Gateway. (The combination of XEP software and the gateway facilities provides you with the complete X.25 native mode programming interface.) The DECnet Router/X.25 Gateway is directly connected to the PSDN and translates the XEP messages it receives into X.25 native mode that allows you to communicate with DECnet and non-DECnet systems over the PSDN.

NOTE

Multiple DECnet nodes can share a single X.25 connection to a PSDN through the gateway or by using DLM.

8.1.4 CCITT Recommendations X.3, X.28, and X.29

Thus far, discussion in this chapter has involved computers or terminals that send "packets" of data to the PSDN. These systems are referred to as packet mode DTEs. There are, however, nonpacket mode systems, such as asynchronous terminals. These systems can also communicate as DTEs over a PSDN. CCITT has defined three recommendations describing the PSDN interface for nonpacket mode: X.3, X.28, X.29.

- **X.3.** Describes the basic functions and user-selectable facilities of the packet assembly/disassembly facility (PAD) for terminals operating in nonpacket mode in a public data network.
- **X.28.** Specifies how to control the user-selectable functions of the PAD from the terminal.
- **X.29.** Specifies procedures for the exchange of control information and user data between a packet mode DTE and a packet assembly/disassembly (PAD) facility.

A PAD is a facility provided by the PSDN. It acts as an interface between the nonpacket mode DTE and the network DCE. The terminal user connects to the PAD and commands it to establish a virtual circuit to a destination DTE. The PAD receives the character-by-character data generated by the user terminal, re-forms it into packets, and dispatches the packets through the network to the remote computer. At the remote computer, a pseudodevice called the X.29 Virtual Terminal Driver handles information sent by the terminal user in much the same way as a regular DECnet terminal device would. That is, it sets terminal characteristics specified by the remote terminal, such as type-ahead options and line terminators and delivers the information to the appropriate application on the remote computer. The PAD and the X.29 Virtual Terminal Driver operate in a manner that is transparent to the user. The terminal appears to the terminal user as though it were directly connected to the remote computer.

Using this same PAD principle, asynchronous terminals connected to a DECnet–VAX node with VAX–11 PSI or VAX–11 X.25/X.29 XEP installed can initiate a circuit with another DECnet or non-DECnet system over the PSDN without buying the services of the PSDN PAD facility. VAX–11 PSI and VAX–11 X.25/X.29 XEP each contain a program that emulates the functions of a PAD on the DECnet system. (The standard DECnet terminal interface is used in these communications.)

In addition to the cost savings entailed in having a built-in "PAD" as opposed to leasing the option from a PSDN, VAX–11 PSI and VAX–11 X.25/X.29 XEP software enable a DECnet–VAX node to initiate and to respond to nonpacket mode communications. Without the VAX–11 PAD function, an asynchronous terminal connected to a DECnet–VAX node can only respond to messages initiated by terminals connected to the PSDN PAD.

The relationships that occur between X.3, X.28, X.29, and the PSDN are shown in Figure 8–2.

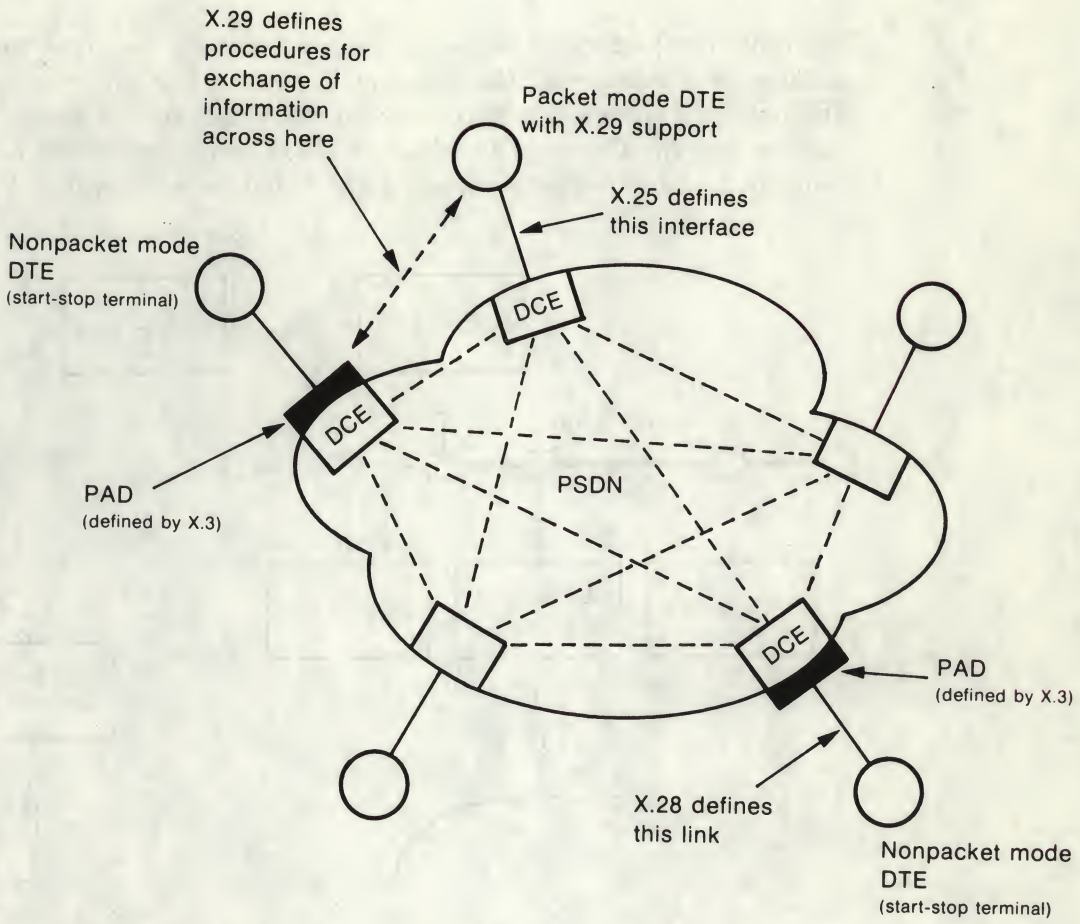


Figure 8–2: Relationships between X.3, X.28, X.29 and the PSDN

The X.3, X.28, and X.29 recommendations are layered on top of X.25. That is, they provide standards for nonpacket mode systems to communicate over a PSDN that are required *in addition to* the protocols defined by X.25. Digital PSI products include provisions for X.25, X.3, X.28, and X.29 recommendations. Therefore, information in this chapter describing X.25 communications, including access methods, also applies to these other standards.

8.2 DECnet/SNA Communications

DECnet/SNA communications involve sending data from one or more DECnet–VAX or DECnet–RSX nodes through a Gateway to an IBM system in an SNA network. The protocol differences between the IBM network and the Digital network are resolved by interceding functions in the Gateway. The Gateway is seen by IBM as a PU (Physical Unit) Type 2 cluster controller.

NOTE

The term Gateway is used in this discussion to refer to all implementations of the DECnet/SNA Gateway node.

The functions performed at the Gateway are not visible to the end user. As far as the user is concerned, the Gateway is a black box that manages to receive DECnet data on one side and send out SNA data on the other. The DECnet node(s) and the Gateway involved in these communications can be located either in a wide area network or on the Ethernet as shown in Figure 8–3.

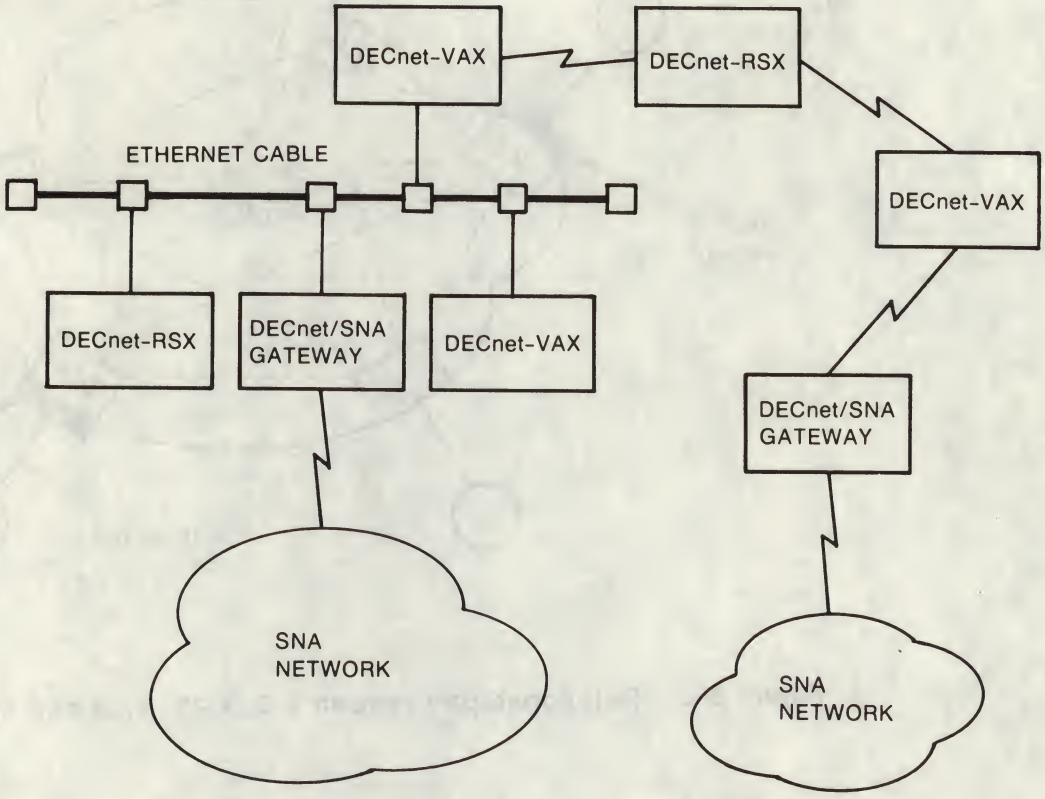


Figure 8–3: DECnet/SNA Environments

The major criteria for participation in DECnet/SNA communications is that you have a DECnet–VAX or DECnet–RSX node with appropriate DECnet/SNA software installed and access to a Gateway. (Operating requirements for the IBM system to communicate with the Gateway, such as supported subsystems and component parameters are detailed in the DECnet/SNA Gateway Installation Guide.)

The software required on the Gateway is specific and unchangeable as described in the DECnet/SNA documentation. You do have some options, however, as to the DECnet/SNA software packages you install on your DECnet system. The packages you install will depend on which functions you want to implement. DECnet/SNA software is available to perform each of the following activities:

- Write application programs that communicate with programs in an IBM network (DECnet/SNA Application Interface software)
- Access IBM applications using a Digital terminal as though it were an IBM 3270 Information Display System (IDS) terminal (DECnet/SNA 3270 Terminal Emulator software)
- Submit jobs to an IBM system and receive job output for them (DECnet/SNA Remote Job Entry (RJE) software)

There are also management tools that enable you to manage and maintain Gateway operations.

8.2.1 DECnet/SNA Circuits

Communications between a DECnet node and the Gateway node occur over DDCMP or Ethernet circuits (see Section 2.2). Communications between the Gateway node and the IBM system occur over SDLC circuits. To perform certain DECnet/SNA functions, you may have to identify particular circuits. Circuit IDs are specified differently for DDCMP and SDLC circuits. Information on specifying circuit IDs of each type of circuit is contained in the DECnet/SNA Gateway management guide for your system.

It is also important to note that some system management activities related to SDLC lines and circuits require you to set the Gateway node as the command executor. Information on how to do this is also contained in the DECnet/SNA management guide for your system.

8.3 Loading the Gateway Node Software

As Figure 8–3 shows, there are two implementations of the Gateway node: one resides on an Ethernet cable, the other does not. Functionally, both nodes are the same. However, if the Gateway is attached to an Ethernet cable, it must be down-line loaded from a DECnet–VAX or a DECnet–RSX host node also on the Ethernet. If the Gateway node is not located on the Ethernet, it can be loaded directly from floppy disks and bootstrapped either manually or by means of a TRIGGER command from a host node.

Chapter 9
How DECnet Supports Applications

This chapter describes how the DECnet capabilities discussed in the preceding chapters support applications running in a user environment.

For illustrative purposes, the user environment posed here consists of several interacting application subsystems that support the daily operations of a fictional company called Fictional Corporation. The subsystems and applications used in this example are:

Marketing Subsystem

- Order Entry
- Sales Analysis
- Market Research

Manufacturing Subsystem

- Production Scheduling
- Inventory
- Quality Control
- Purchasing

Accounting/Financial Subsystems

- Billing
- Accounts Receivable
- Short-term Cash Management

Figure 9–1 illustrates the interaction among these subsystems and applications only insofar as the following discussion is concerned. The financial subsystem and the short-term cash management application, for example, interact with other subsystems and applications not shown here.

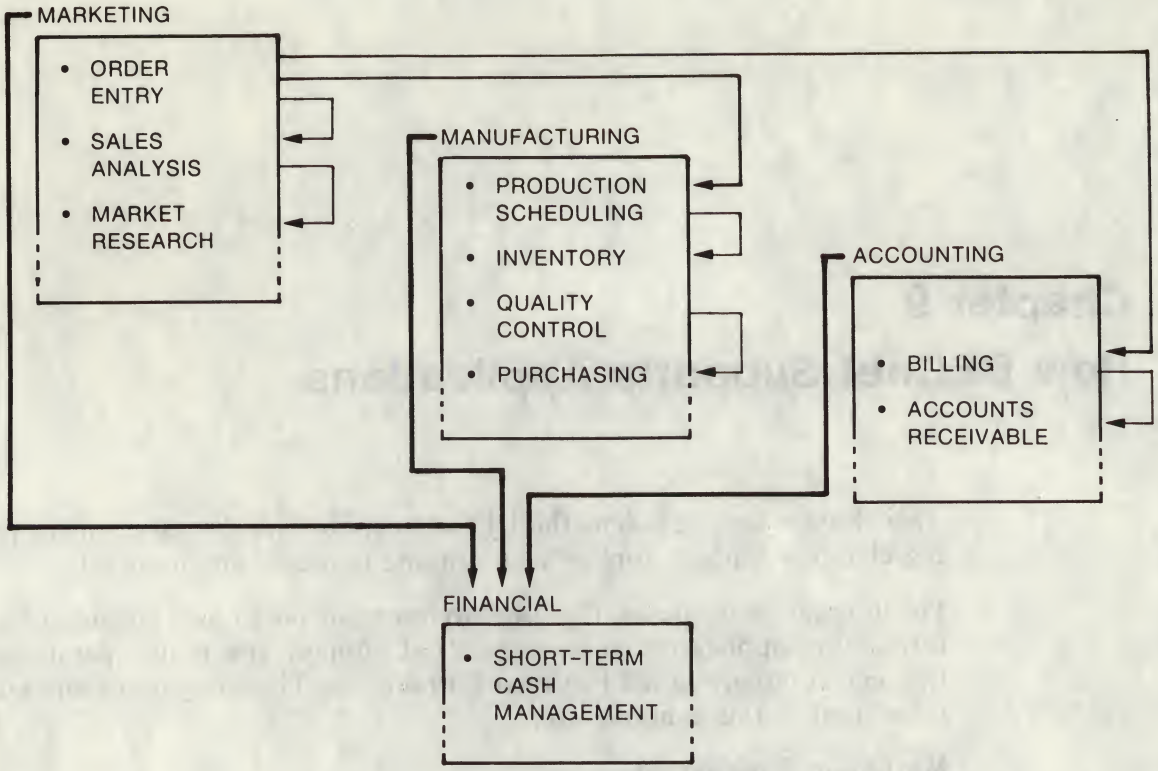


Figure 9–1: Subsystem and Application Interaction at Fictional Corporation

The order entry application captures a basic set of information, including customer name, address, products ordered, and delivery requirements. If products ordered come out of finished goods inventory, the basic set of information is then input to order assembly and dispatch applications; if the ordered goods are to be produced, as is the case in the example to be presented, the information goes to a production scheduling application, which then interacts with raw materials inventory and/or component inventory. If required, the inventory applications interact with purchasing. The production scheduling system updates a corporate data base that records historical and scheduled utilization of production facilities.

The billing and delivery scheduling applications, not shown here, also use the basic information set, along with other information retrieved from various data bases accessed and updated by the complex of application systems. Accounts receivable, which interacts with credit and billing applications, is updated on a regularly scheduled basis.

Since Fictional is an international company, its application programs must interact both within and over national boundaries. The nodes in Fictional's DECnet network communicate over national packet-switching networks in the United Kingdom, the United States, Canada, France, the Netherlands, and West Germany. These cost-effective communications facilities implement the X.25 recommendation of the CCITT. Requirements for communicating over such PSDNs within and over national boundaries are met at Fictional by means of the DECnet Router/X.25 gateway nodes and X.25 Packetnet System Interface (PSI) capabilities resident in general purpose DECnet nodes (see Chapter 8).

Company structure dictates a need for combined wide area and local area networks. The processors, terminals, and personal computers connected to the network are at functionally defined locations, such as sales offices, manufacturing facilities, warehouses, distribution centers, and corporate headquarters. DECnet nodes at these locations are linked to one another by leased lines and X.25 networks. In addition, some applications require interaction between DECnet nodes and IBM/SNA systems.

The following sections illustrate how DECnet capabilities support Fictional's applications. DECnet capabilities discussed in this context include:

- Task-to-task communication
- File Transfer
- Remote File Access
- Terminal Communications

9.1 DECnet's Task-to-task Capability in an Application Environment

DECnet's task-to-task capability handles all the network processing that enables Fictional's application systems to interact with one another over the network. As noted in Chapter 5, this same capability also enables application programs to interact with DECnet modules and DECnet modules to interact with one another. In this example, only the program-to-program interaction is discussed.

The only requirement that network communication imposes on Fictional's programmers is that the applications issue calls to network services and that they be written to cooperate with one another insofar as data formats and function-specific data transfer operations are concerned. Logical link processing by DECnet is completely transparent to terminal operators.

The logical links (see Chapter 4) over which the numerous applications communicate are set up automatically by the DECnet software. Because it is designed in accordance with the specifications of the Digital Network Architecture, the DECnet software offers a standard application-network interface. All of Fictional's applications, no matter in what DECnet nodes they run, access the network by means of the same mechanism. This common approach to the network interface encourages open-ended design of application systems. This, in turn, assures that future applications developed by Fictional's programmers will be compatible with the existing set at the networking level.

9.1.1 Order Entry and Production Scheduling

As an example of how DECnet task-to-task capabilities operate, the following discussion outlines the environment in which logical links are set up, controlled, and terminated as the order entry and production scheduling applications interact.

The order entry application runs in local sales offices. Figure 9–2, which isolates one area of the corporate network, illustrates three levels of communication:

- Between local and regional sales offices (order entry application and sales analysis application)
- Between regional sales offices and corporate headquarters (sales analysis application and market research application)
- Between local sales offices and corporate headquarters (order entry application and production scheduling application)

Basic information captured at the local level consists of customer name, billing and delivery addresses, products ordered, and delivery requirements.

The production scheduling application is centralized at corporate headquarters. This application system matches order requirements against available production facilities (as recorded on the production scheduling data base) and against raw materials and component inventory. It then posts an indication back to the order entering sales office as to whether or not delivery requirements can be met. If they can be met, the production scheduling system notifies the appropriate manufacturing locations and updates the production scheduling data base.

9.1.2 Network/Application Interaction

Logical link processing between the applications covers three distinct operations:

- Establishment of the link
- Control of the link
- Destruction of the link

Each is discussed in the following.

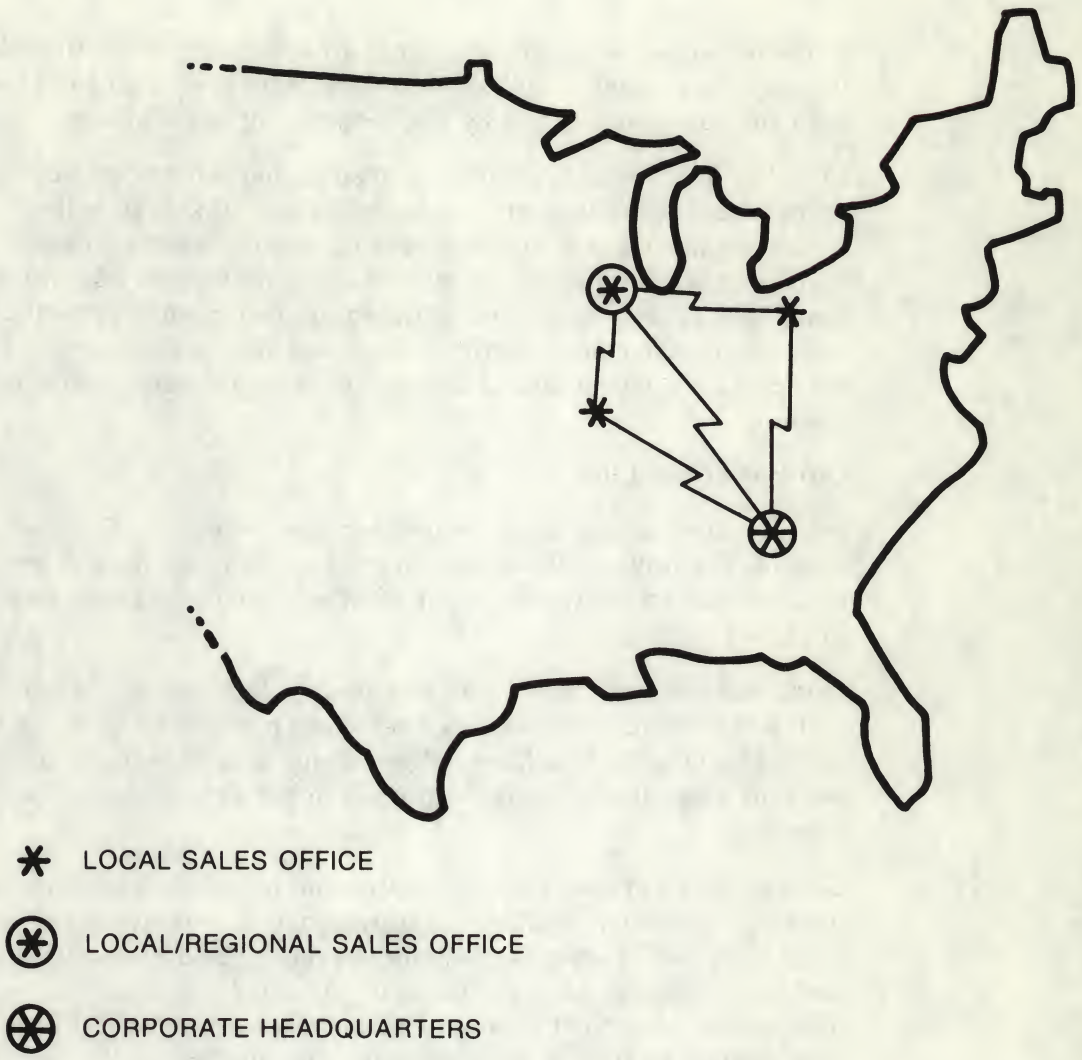


Figure 9–2: Three Applications Linked by DECnet's Task-to-task Capability

Establishment of the Link

Data entry operators working with VT100 terminals at the sales offices invoke the order entry transaction system and complete order entry forms displayed on the terminal screens. From the operator's viewpoint, the procedure is straightforward. Order information is entered on the forms; the completed forms are ENTERED; a new form appears. Transmission is transparent. On an applications level, the entered orders are checked locally against customer and product files. Any errors are indicated back to the operator for correction.

On a network level, similarly transparent to the operator, the order entry system opens communication with DECnet. Then, by issuing connect requests, it tries to establish logical links with a sales analysis system at the regional sales office and with the production scheduling system at corporate headquarters.

If the other applications are ready to communicate with order entry, the programs are logically linked, and they begin exchanging data in accordance with the rules established by the cooperating applications.

Both the production scheduling and sales analysis systems could be receiving order entry information simultaneously from other terminals at the same and at other sales offices. DECnet sets up unique logical links for each pair of applications bound in a session. Any communications from production scheduling back to order entry during the same session, such as notification that the delivery requirements specified cannot be met, are transmitted over the logical link initially established between the two programs at the beginning of the session.

Control of the Link

In the course of the sessions between the programs, DECnet monitors and controls the links, making sure that all transmitted data is received and that all data received at the data link level is turned over to the receiving program in proper sequence.

Order data records are of variable length. DECnet at the sending node segments the record and specifies a sequence number for each segment. DECnet at the receiving node acknowledges receipt of each segment by sequence number and assembles message segments in the order specified by the sequence numbers.

To make certain that the order entry and production scheduling applications interact efficiently, DECnet requires that the receiving program issue RECEIVE calls that set up buffers for the receipt of data. DECnet at the sending node will not transmit data unless it is notified that a buffer is available at the destination node for the receipt of message segments. DECnet employs this flow control technique to make sure that buffer overflow conditions do not arise. Conditions, therefore, that cause messages to be retransmitted, thus increasing communications overhead, are not permitted to arise.

Destruction of the Link

When all order data has been entered, DISCONNECT and CLOSE calls are made to the network. The logical link binding the programs is destroyed, and the network resources that were employed in maintaining that link are released.

9.2 DECnet's File Transfer Capability in an Application Environment

At Fictional Corp., file transfer operations over DECnet are initiated by terminal operators and by programs. Examples of each are presented in the following subsections.

9.2.1 Market Research and Sales Analysis

Fictional's market research system runs at corporate headquarters. It takes input from the sales analysis applications running at the regional sales offices and from market research analysts located worldwide (see Figure 9–3).

Input from sales analysis provides historical information that is used to plot product acceptance trends. Input from market research analysts quantifies market trends in specific geographical areas and projects the impact of new products on existing and forecasted customer bases.

Market research analysis is supported by a series of charts produced by the sales analysis application. These charts are written to files that are then made available to the research analysts. Terminal operators at the regional sales offices use electronic mail over DECnet to notify the analysts when a new set of charts is available.

The analysts edit these charts, supplying additional input and modifying existing data. Some use DECnet file transfer capabilities to transfer the chart files from the market research application at corporate headquarters to their local nodes; others use the same DECnet facility to copy them to accounts that they have on the node running the market research system at corporate headquarters. Some prefer to print the files and work on the hard copies of the charts; others use the editing feature available at their local nodes to update the charts on-line.

Analysts working at DECnet–VAX nodes, for example, use the COPY command, as follows, to obtain the chart files. Note that in this and in subsequent examples, user input is printed in red and system output is printed in black.

```
$ COPY CVAX::CHART.CUR *.*
```

In the above example, the research analyst copys a file called CHART.CUR that resides on node CVAX. This command line shown here causes CHART.CUR on node CVAX to be written to a file of the same name on the local node. A different file name can be specified.

As shown in Figure 9–4, the COPY request (solid line) and response (dotted line) may be routed through multiple nodes. To the requester, however, this network activity is transparent. The preferred path for each transmission is selected by the DECnet routing capability.

The DECnet–VAX node at corporate headquarters runs ALL-IN-1 software, an integrated set of office applications that includes word processing, calendar and desk management, graphics, mail, spread sheet, and other capabilities. Analysts with accounts on this node frequently use the integrated spread sheet, graphic, and word processing capabilities to prepare reports and presentation material. These can then be either sent to or made available over DECnet to others with proper access to these reports. The timeliness with which these reports are prepared and distributed enables management to react to market shifts and adjust production, development, and advertising direction appropriately.



Figure 9–3: Market Research Application Takes Regional and Worldwide Input

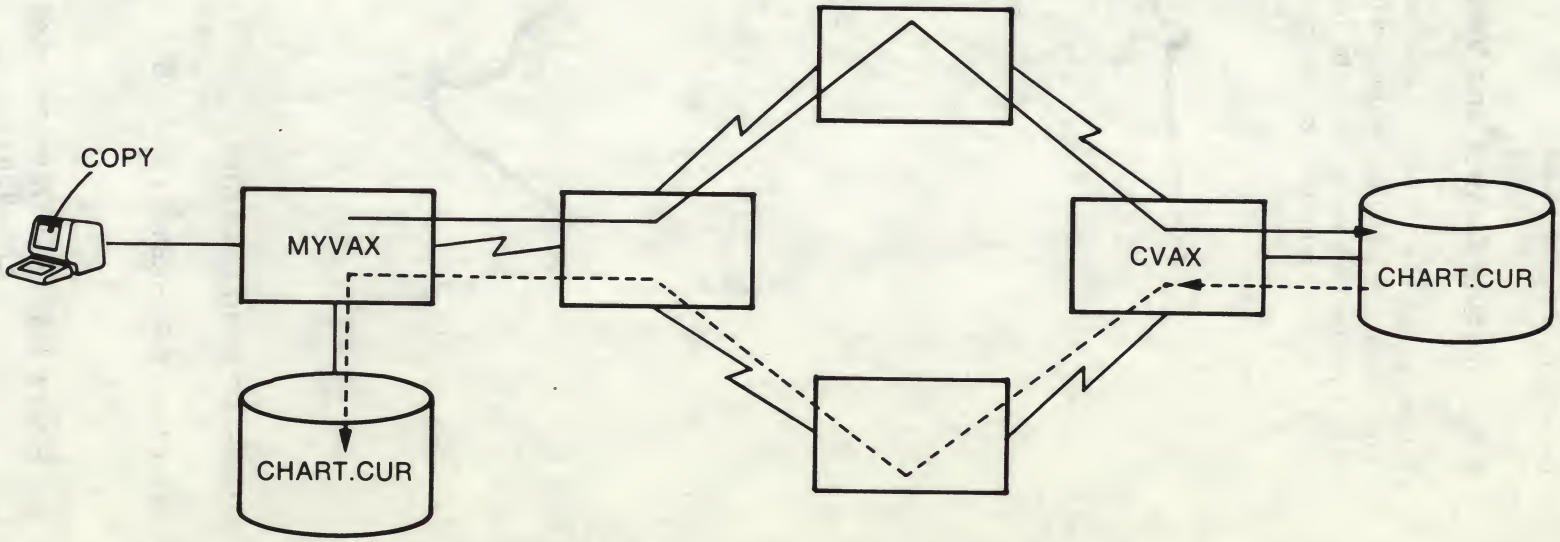


Figure 9–4: COPY Request/Response Is Shown Routed through Multiple Nodes

The transfer of the files to or from the research analysts' systems uses the DECnet mechanisms described in Chapter 6. As noted, operation of these mechanisms is completely transparent to the terminal users. Programs that handle file transfer and reception, as described below, must include proper calls to network services.

9.2.2 Quality Control and Vendor Analysis

Programmed file transfer operates at Fictional in the interaction between quality control and purchasing systems. The quality control applications run at the company's manufacturing locations, and the purchasing systems operate at regional purchasing centers (see Figure 9–5).



Figure 9–5: File Transfer between Manufacturing and Purchasing Locations

In one of the quality control (QC) operations, incoming components used in product assembly are screened by sensors that analyze dimensional characteristics. The sensors are multidropped from DECnet–RSX nodes that control sensor operation and that translate sensor input to formatted data.

Results of quality control analyses are accumulated by component/shipment/vendor/QC location. On a daily basis, a program run in a DECnet–VAX node at each of the plants accesses this information and builds files by manufacturing location/date/vendor/shipment/component/QC location. It also develops information on rejected items in terms of units and dollar value for each vendor/shipment/component and for each QC location.

On third shift, when systems and lines are less busy, a data acquisition program in the purchasing subsystems uses the DECnet file transfer capability to move these files to the regional purchasing centers, where the data is input to a vendor analysis and evaluation program.

9.3 DECnet's Remote File Access Capability in an Application Environment

The file transfer capability, as described in the preceding section, is a subset of DECnet's overall remote file access capability. This capability enables Fictional programs and users to transfer files from one node to another, to manipulate files on remote nodes, and to display remote files at local nodes. The DECnet mechanisms used to perform these operations are identical to those used in file transfer.

9.3.1 Network Virtual Terminals

Another DECnet feature that supports extensive remote file access operation at Fictional is the network virtual terminal capability. This capability enables a user at a terminal at one node to log on to a remote node and operate as if the terminal were directly attached to that node. Users of this capability must, of course, have proper access to the remote nodes and files.

Although the virtual terminal capability is exercised slightly differently for each DECnet implementation, overall procedures are similar. The following example shows how a user at a terminal on a DECnet–VAX node would use the SET HOST command to log on to a remote node to use resources located there.

```
$SET HOST ATLANTA
Username:RAINER
Password: (not echoed)
Welcome to VAX/VMS Version 3.4 on node ATLANTA
$
```

In the above example, a user, identified as RAINER, at a terminal connected to a DECNET-VAX node, entered a SET HOST command to log on to node ATLANTA. See Figure 9-6. The system then asked RAINER for his name and password. The welcome message that printed out indicates that RAINER has a valid account on node ATLANTA. The prompt ($) that follows, shows that RAINER is logged on to the node and can use its files and other resources. If RAINER did not have a valid account on that node, an "Unauthorized User" message would print out and RAINER would be returned to his local system.

The following section describes an application development project currently underway at Fictional. This project is expected to optimize the corporate treasurer's ability to identify and invest idle funds using DECnet's remote file access capability.

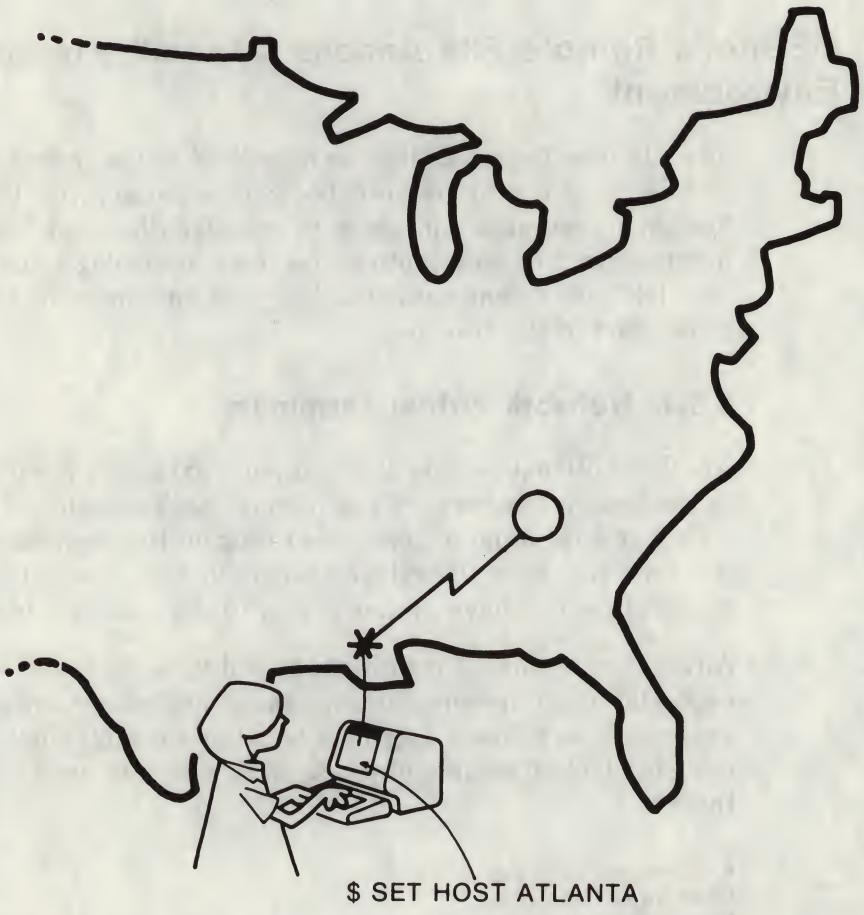


Figure 9-6: Local User Logs onto Remote Node

9.3.2 Short-term Cash Management

Corporate funds are kept in many accounts at foreign and domestic banks. A short-term cash management application system currently being developed at Fictional will enable the corporate treasurer to check the status of these funds, identify idle cash, and then transfer these nonworking assets to the portfolio management group for investment, specify the funds for debt reduction, or retain them in the current account.

DECnet's remote file access capability is a principal component in this projected capability.

The application development effort is concerned with developing appropriate security measures, identifying parameters and processing requirements that will help define "unencumbered funds," with cross-referencing regional data bases for relevant cash and local currency value data and with developing algorithms that equate investment opportunities with available cash.

Using the communications capability provided by Gateways, and by DECnet's remote file access capability over local area, wide area, X.25, and SNA networks, the treasurer's office will be able to isolate and examine cash records maintained in regional data bases worldwide. Analysis of this information will then enable the treasurer to make timely decisions as to transfer and investment actions.

9.4 DECnet's Terminal Facilities in an Application Environment

DECnet's terminal facilities include:

- Interactive terminal communications functions integrated into the network software
- Functional capabilities that are given broad distribution through DECnet's communications capabilities
- Other capabilities based on DECnet terminal configuration options

Interactive terminal communications, as described in Chapter 7, enable users to communicate with one another over the network in real time.

DECnet's communications capabilities enable terminal users to access locally resident or remote processing capabilities and to distribute the output over the network. In other words, any node in the network can be used as if it were the local node.

Configuration options enable users to incorporate general purpose node-based or terminal server-based network virtual terminals into their networks. As described in Section 7.2, the virtual terminal capability enables you to log on to any supportive node in the network and use the resources of that node. Further, the ability to include personal computers in a DECnet configuration and to use them in both processing and terminal emulation modes adds a dimension to the terminal facilities available through DECnet.

The following sections summarize some of the ways in which Fictional Corporation uses these capabilities in its daily operations.

9.4.1 Order Entry and Production Scheduling

Production scheduling frequently sees some on-line communication between a sales office and a manufacturing site. If, for example, the sales office is notified by the production scheduling system that customer delivery requirements cannot be met, the sales office might consult with the customer and negotiate another acceptable date for delivery of ordered products.

A terminal operator or the salesperson will then invoke a DECnet interactive terminal facility (Phone or TLK, described in Chapter 7) at a sales office terminal and communicate revised delivery instructions to the manufacturing location.

When working at a terminal on a DECnet–VAX node, for example, the salesperson would invoke the Phone utility. See Figure 9–7. In the example that follows, a dialog is established with a user at another DECnet–VAX node at a manufacturing location named MFG7.

```
$ PHONE MFG7::SCHED
```

This command places a call to a user named SCHED at the manufacturing site. If SCHED is currently logged on to the DECnet–VAX node, a message will be displayed indicating that a message has been received. If, for example, the sender is a user named GREEN at a node called SALES4, the message will read as follows:

```
SALES4::GREEN is phoning you on MFG7
```

To receive and respond to the call, SCHED enters:

```
$ PHONE ANSWER
```

This reply causes both terminals to display a split screen. The top half of the screen displays the local user's input and the bottom half displays the remote user's input. Each party to the dialog can enter text at the same time.

After using this facility at Fictional to arrive at a mutually acceptable schedule for production and customer delivery, the manufacturing location specifies it back to the production scheduling data base at corporate headquarters.

Fictional finds that informal interaction outside the applications frequently resolves such customer service issues more quickly than do sets of systematized procedures. DECnet interactive terminal communications support such activities throughout the network.

9.4.2 Market Research

Because effective market research must be responsive to current market trends, market research analysts at Fictional develop forecasts and test new product marketing strategies in real time. Field input is played against market models; results are analyzed by statistical processing capabilities, and "what if" input is developed to refine and extend the models.

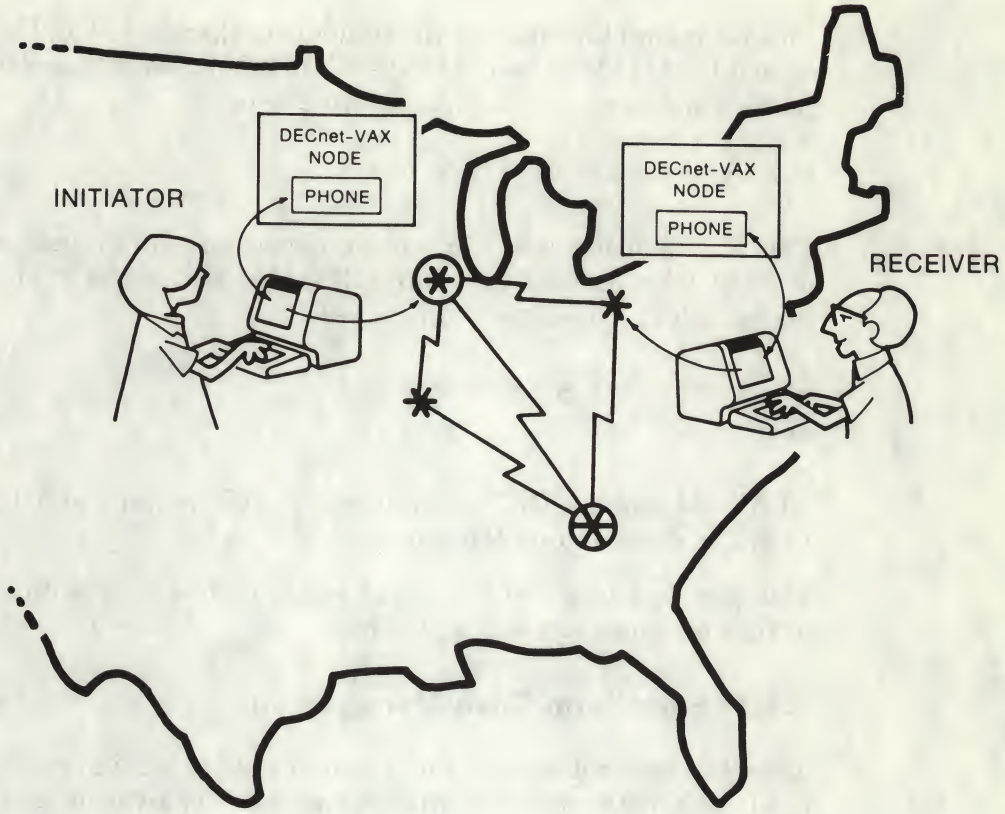


Figure 9–7: Phone Capability Permits Rapid Communication between Users

Research analysts use DECnet's terminal facilities extensively in performing this work. Field analysts provide input via VT100 terminals connected to nodes at regional locations. Analysts at corporate headquarters use VT125 terminals connected to a DECnet–VAX node with node-resident ALL-IN-1 software capabilities to develop reports and presentation material.

In report development, the analysts use the integrated ALL-IN-1 software capabilities to combine spread sheet computation, graphics development, and word processing. When required, the mail capability of ALL-IN-1 is called up to distribute reports and analyses over the network.

In addition to the integrated mail capability provided by ALL-IN-1, several similar capabilities are available to DECnet users who might not need all of the capabilities of the integrated package. The VMS MAIL capability, for example, enables users at DECnet–VAX nodes to exchange correspondence.

To send a message, the user simply enters the MAIL command, specifies the SUBJECT of the correspondence, and either specifies a file to be sent or types the message in response to a prompt. For example:

```
$ MAIL/SUBJECT=ZONE12 UPDATE.MKT LONDON::GRAHAM
```

This command line delivers the contents of the file, UPDATE.MKT, to a user named GRAHAM at node LONDON. If GRAHAM is logged on to node LONDON, a message such as the following appears:

```
New mail from YORK::PEEK
```

This line indicates that the waiting message is from a user named PEEK at node YORK. To read the mail, GRAHAM enters the MAIL command and presses RET. A message is displayed:

```
You have 1 new mail message
MAIL>
```

GRAHAM presses RET in response to this prompt and the mail sent by PEEK is displayed on the screen.

Mail sent to a terminal that is not logged on to a node is indicated to the user of that terminal at the next log in.

9.4.3 Short-term Cash Management

The PRO/DECnet node in the treasurer's office is connected with an Ethernet local area network cable that serves the department and other adjacent groups. This personal computer serves the treasurer in several ways:

- As a stand alone processor with its own self-contained file and disc services
- As a node on the DECnet network, capable of interacting with computing capabilities and other resources located elsewhere on the network
- As a terminal on the DECnet network that can access other nodes and use resources located at those nodes

The network mail services available to the treasurer through this node or through another host node on the network will be of significant value when the short-term cash managment application is fully implemented.

Decisions to use available cash must take into account both internal and external factors in a very dynamic environment. Internal factors are represented by such considerations as payables, operating requirements, debt constraints, and so forth. External factors (such as bank policies in regard to minimum balances, for example) are important considerations in determining thresholds that define cash as excess. Such policies, however, are frequently conditioned by the general economic climate, government regulations in regard to fund transfer, as well as other uncertainties that govern international financial transactions.

Using the mail and dialog facilities, the treasurer can keep informed, in real time, of actual or potential changes in such conditions. Fund accumulations can be monitored along with those conditions that can affect movement of those funds.

Records of information that influence investment decisions can be retained on line for quick lookup and can also be printed out if hard copy is desired. This kind of information will enable the treasurer to optimize decisions in regard to existing and potential inventories of investable funds.

9.5 DECnet as a Consideration in Relocation Planning

In planning relocation of its corporate headquarters, Fictional Corporation management, along with a group of outside consultants, carefully examined the consequences that the projected move could have on future operations.

The company planned to move to an exurban location, outside the commuting distance of a major metropolitan area. Real estate costs and tax advantages were attractive; a dependable labor pool was available; and the projected location was also desirable in terms of cultural and recreational activities for employees.

The expense to be incurred, however, was significant. Architectural fees and construction costs, along with the outlay required to relocate existing employees who decided to go along with the move, represented a very sizeable investment. Major questions arose concerning the threat posed to this investment if, in the future, a decision was made to change the company's structure. Would the physical shell into which they planned to move threaten to define the logical structure from which they would continue to operate, even if future business considerations dictated change?

What, for example, would happen if Fictional decided to sell a subsidiary operation, or to decentralize certain operations that were now centralized, or to realign intergroup and reporting relationships? Management realized that in moving from its current urban location, Fictional would be cutting itself off from the numerous services — such as availability of rental space and a market for subleasing — that are common to a metropolitan environment. Would the new physical plant in its proposed location — and built at great expense — be flexible enough to continue serving a changing organization?

Together with the consultants, Fictional management determined that recent developments in computer technology, such as those offered by DECnet, are new and positive elements to be considered in dealing with such questions. Local area networks, for example, permit those changes in information flow that accompany reporting and administrative realignment to be put in place rapidly, without disrupting ongoing business. Meanwhile, communication with facilities anywhere in the world continues to use network facilities that are already in place.

In dealing with these matters, Fictional learned that investment in relocation is afforded a degree of protection by flexible and responsive computer communications facilities, such as DECnet, even through changes in business structure. Computer networking, it was determined, provided a measure of mobility to capital investment.

SYSTEM MANAGEMENT

Chapter 10
Network System Management

This chapter outlines the network-related tasks of a DECnet system manager and describes the facilities that DECnet provides to perform those tasks. In so doing, the chapter also provides many definitions of network entities and parameters that pertain to network system management. In practice, many people may be responsible for carrying out the described tasks. For simplicity, these people are collectively defined by the job title "system manager." Those who manage network nodes are responsible to both local and remote users and should be aware that while local applications usually demand the greatest share of system resources, the remote users, a potentially very large group, must be sure of each node's response to network applications.

The procedures for performing network management functions can vary from system to system. For example, the procedure for generating a DECnet/E node is different from the procedure for generating a DECnet–RSX node. RSTS/E is a time-sharing system with a software structure quite unlike the real-time, event-driven RSX–11M. Because each DECnet implementation (DECnet/E, DECnet–VAX, and so on) is an extension of an operating system, the different ways to manage a node reflect the differences between the basic Digital operating systems. The purpose of this chapter is to provide an overview of network management functions rather than to describe actual procedures for performing them. Procedural information can be found in the documentation provided for a specific implementation.

Network system management functions in DECnet include:

- **Planning for node generation** (see Section 10.2). This process tailors DECnet software to suit a specific node's network application. Node generation planning should be conditioned by an overall network plan that establishes basic network-wide conventions. Implementation of such conventions will assure communications and processing capability throughout the network.
- **Generating network software** (see Section 10.3). This section discusses building the tailored DECnet software to create an active node.
- **Defining and redefining network parameters** (see Section 10.4). This section introduces various network parameters whose definitions determine many aspects of a node's role within a specific DECnet configuration.

- **Operating a node** (see Section 10.5). This section discusses operational functions such as starting up and shutting down a node and the physical lines connected to a node.
- **Monitoring node activity** (see Section 10.6). This section discusses monitoring the day-to-day performance of a node by gathering and analyzing logging data that DECnet makes available.
- **Down-line loading** (see Section 10.7). This section discusses the procedure for loading system and task images from an executor node to satellite nodes such as RSX-11S and communication server nodes.

Network system management responsibilities also include the testing and monitoring functions described in Chapter 11.

10.1 Network Management Utilities

Network management utilities are the means by which a system manager performs most of the functions described in this chapter. All DECnet implementations support one or more utilities that provide access to DECnet management modules. These modules perform the functions defined by the Network Management layer (see Section 1.3.1). All implementations support a utility called the Network Control Program (NCP).

Table 10–1 briefly describes the function of NCP and other network management utilities that each DECnet implementation uses. Each utility accepts commands that activate DECnet management modules either to perform specific tasks or to request information about the current state of the local node or the network. Each utility is described in detail in the system manager's or user's guide for each DECnet implementation.

10.2 Planning for Node Generation

Planning for node generation entails gathering and consolidating information. Digital-supplied DECnet software provides generalized network capabilities, but the user must supply the data and programs that create a live DECnet application. A system manager accumulates these data and programs for eventual incorporation into the local node.

Each system manager must ensure that managers elsewhere in the network receive information that they need about other nodes. Programmers responsible for network applications should cooperate with system managers by exchanging information. For example, programmers must use correct addresses in calls that generate connect requests (see Section 5.1), while system managers must know the correct names and object types that the network programs use to identify themselves.

Table 10–1: DECnet Systems and Network Management Utilities

System	Utility	Function
DECnet-VAX	NCP	Loads, controls, monitors, and tests DECnet software; defines configuration data base parameters; down-line loads DECnet/11S and communication server nodes
DECnet-RSX	NCP	Loads, controls, monitors, and tests DECnet software; down-line loads DECnet/11S and communication server nodes
	CFE	Changes parameters in the configuration file CETAB.MAC, which is produced at network generation
	VNP	Changes the disk image of a DECnet-RSX system (VNP cannot be run from a DECnet/11S node)
DECnet/E	NCP	Loads, controls, monitors, and tests DECnet software; maintains the DECnet/E parameter file; reports the current status of active logical links, of known physical lines and remote nodes, and of programs using local and remote send/receive services
DECnet-IAS	NCP	Loads, controls, monitors, and tests DECnet software; defines configuration data base parameters
	CFE	Changes parameters in the configuration file CETAB.MAC, which is produced at network generation
DECnet-RT	NCP	Loads, controls, monitors, and tests DECnet software; defines and changes configuration data base (CETAB.MAC) parameters
DECnet-20	NCP	Loads, controls, monitors, and tests DECnet software; defines the configuration data base; down-line loads system and task images
DECnet-10	NCP	Loads, controls, monitors, and tests DECnet software; defines the configuration data base; down-line loads system and task images

Legend:

CFE — Configuration File Editor
NCP — Network Control Program
VNP — Virtual Network Processor

10.2.1 Configuration Data Bases

Every DECnet node has some form of configuration data base that defines characteristics of the local node and determines how that node functions within the network. In some cases, Digital-supplied software already includes such a data base to provide initial default values for many data base entries. For other implementations, the network generation procedure creates the data base. Table 10–2 lists the term that each DECnet implementation uses to identify its configuration data base.

Depending on the type of DECnet node, the configuration data base may need to be updated periodically to reflect changes in the network or to tune the performance of the network. Section 10.4 explains many of the parameters typically included in a configuration data base. The facts and figures needed to define them must be gathered before a node can actively participate in a network.

Table 10–2: Configuration Data Base Terms

System	Term	Comments
DECnet–VAX	Configuration data base	The initial data base is provided within the DECnet software supplied by Digital. NCP commands are subsequently used to modify the data base.
DECnet–RSX DECnet–IAS	Configuration file	This file (CETAB.MAC) is created during network generation and subsequently can be modified by the Configuration File Editor (CFE).
DECnet/E	Parameter file	This file ($NETPRM.SYS) is created and subsequently modified by NCP commands.
DECnet–RT	Configuration file	This file (CETAB.MAC) is created during network generation and subsequently can be modified by network management commands.
DECnet–20	*rev*-CONFIG.CMD	This file is shipped with the TOPS–20 monitor and modified by the installer to define the local hardware configuration.
	nodename.CNF NETPAR.MAC	This is the network configuration file. This file contains system-specific parameters.
	Configuration file	This file (CETAB.MAC) is created during network generation and subsequently can be modified by network management commands.
DECnet–10	Configuration files	These files (*mapping*.CNF and CETAB.MAC) are created during network generation and subsequently can be modified by network management commands.

10.2.2 Network Generation Planning Aid

DECnet–IAS and DECnet–RT each provide an aid to help users plan for node generation. The aid consists of a command file that contains questions pertaining to node generation options. A system manager runs this command file from a terminal and answers the questions according to the requirements of the local node.

Using the system manager's responses, the command file then generates several worksheets. Each worksheet tells the system manager how to generate some part of DECnet to reflect local requirements. See the appropriate network generation manual for a complete description of the command file and the worksheets it generates.

10.3 Generating Network Software

DECnet software arrives from Digital on distribution media such as magnetic tapes or floppy disks. The type of medium depends on the DECnet implementation and, in some cases, on the hardware configuration of a specific system. For example, DECnet–RSX is distributed on several media to accommodate a variety of user hardware configurations.

The procedures for using the distributed software to generate an active node are different for each implementation of DECnet. To generate an RSX or an IAS node, a system manager has to regenerate the operating system first and then tailor and build the network application on top as a second procedure. Other implementations do not require a system manager to rebuild the distributed software.

A specific list of DECnet software modules depends on the implementation and on the specific network application. However, the distributed software can include modules like the ones listed below, many of which have been discussed in preceding chapters.

- Network device controllers
- A DDCMP module
- Ethernet
- Routing layer modules
- An NSP module
- Network utilities like NCP, NFT, and TLK

10.4 Defining Configuration and Other Static Parameters

The parameters that make up a node's configuration data base are relatively permanent or static, because changing them tends to change the way the node functions within the network. In addition to parameters that are strictly part of a configuration data base, a system manager must define other parameters, such as local node network object descriptions (see Section 5.2.1.1), that affect the way a node functions within a network.

Depending on the implementation, configuration and other static parameters are defined in various ways. They can be defined at network generation by means of NCP commands, or they may be predefined in Digital-supplied software. Brief descriptions of static parameters that are typically defined for most types of DECnet nodes are provided in the following subsections.

10.4.1 Node Addresses and Names

The system manager must assign a numeric address that uniquely identifies that node within the network. See Section 2.1 for a discussion of node addresses and node names.

10.4.2 Node Verification Passwords

Whenever one of its physical lines is turned on, a node exchanges initialization messages with the remote node at the other end of the line. (Section 10.5.2 discusses starting up a physical line.) The messages exchange information such as the version numbers of the node's DECnet software modules or the node's type — routing or nonrouting.

After exchanging initialization messages, a node can request that the adjacent node verify its identity by supplying a password. If such verification is required, adjacent nodes must supply passwords to gain access to the local node. If verification is not required, adjacent nodes do not have to supply passwords, and they automatically gain access to the local node after exchanging initialization messages. The access gained or denied at this stage has to do with the ability to send and receive messages over the physical line connecting two nodes.

Within a DECnet configuration that enforces node verification, each node maintains a data base of passwords that it sends to and expects to receive from its neighbors:

- **Receive password.** The password that the local node expects to receive from the adjacent node. If the password actually received does not match the receive password expected, the local node denies access to the adjacent node.
- **Transmit password.** The password that the local node sends to the adjacent node. The transmit password must match the receive password that the adjacent node expects from the local node.

10.4.3 Network Object Parameters

Most nodes maintain a data base that describes all the network objects, both user-written programs and DECnet modules, currently residing in the node and capable of engaging in network activity. (Network objects are described in Section 5.2.1.) The manner in which the node's object data base is maintained depends on each node's implementation. Typically, a node stores the following kinds of information:

- Object types, names, and addresses
- Access control and verification information associated with each object
- The number of copies of a specific object that DECnet can run to satisfy incoming connect requests

10.4.4 Routing Parameters

Several parameters affect the operation of the routing module in Phase III and Phase IV full-routing nodes. (See Chapter 2 for a discussion of full routing.) These parameters establish characteristics such as:

- A maximum path cost that limits possible routes to paths that cost the same as or less than this value.
- Individual line costs that figure in routing algorithms used by the routing module. Each line cost is a number from 1 to the maximum path cost set for the node. Higher line cost can lead to less traffic on the line because the routing modules dispatch packets on the least costly paths. Lowering the cost of a line does not always increase the traffic it carries. For example, if a line leads to an end node, line cost does not affect that line's traffic.

 If one or more of a node's lines actually cost more than others to use, the traffic on those lines can possibly be diminished by raising the routing line costs assigned to them. In fact, this can be done whenever the system manager wants to decrease traffic on a line.
- A maximum number of hops per path. A node is unreachable if it cannot be reached within the maximum number of hops.
- A routing timer that determines the interval between automatic updates of the local node's routing data base.
- A buffer size for the unit of data actually transmitted over physical lines by the transport module.
- A buffer count that determines the size of the routing module's pool of available buffers.

10.4.5 Line Identification

Each communication line leading from a node has a unique identification that is used in commands. A node's configuration data base usually contains the identification of each communication line. This identification has the following general format:

dev-c-u

where

dev is a mnemonic for the type of device — for example:

UNA — the DUENA multiaccess communications link (Ethernet)
DUP — the DUP11-DA
DMC — the DMC11-DA/AR, -MA, AL, or FA/AR
DZ — the DZ11-A or -B

c is a number (0 or a positive integer) designating the device's hardware controller.

u is a unit/line number (0 or a positive integer) included if the device is a multiplexer.

Examples of line identification follow:

Identification	Description
UNA-0	DUENA, controller 0
DMR-0	DMR11, controller 0
DZ-1-0	DZ11, controller 1, unit 0

10.4.6 Circuit Parameters

In addition to identifying actual physical lines, most DECnet implementations also identify circuits, which are logical point-to-point communications paths. At nodes with these implementations, the circuit rather than the line is manipulated and defined to control the flow of data between nodes. Physical lines become the medium over which circuits operate. As a reflection of this concept, DECnet–VAX, DECnet–RSX, DECnet–20, DECnet–10, and DECnet/E users frequently specify circuits when users at other DECnet nodes specify lines in equivalent network management commands.

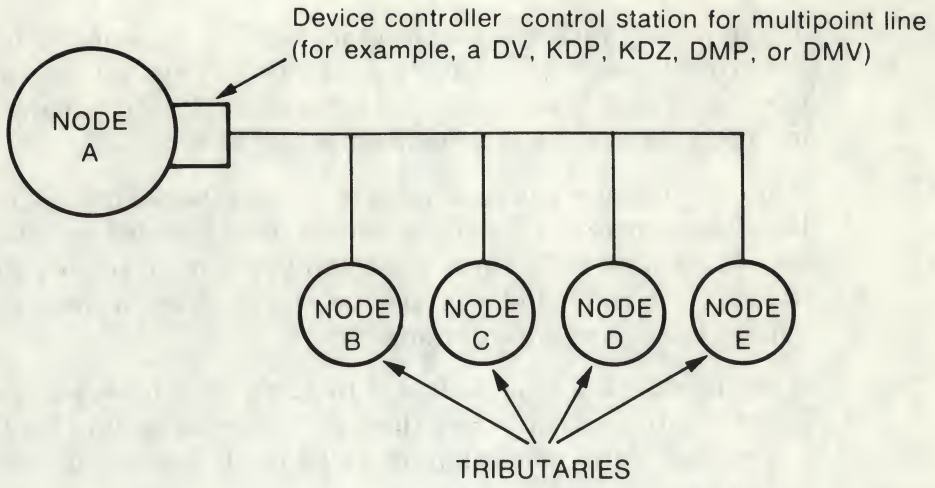
Circuits that handle DECnet traffic correspond closely to the physical lines that actually transmit the data. When a circuit corresponds to a point-to-point line, circuit IDs and line IDs are exactly the same. For example, the string DMC–0 can identify either the line or the circuit in a network management command. However, circuit and line identifications associated with a multipoint line and its tributaries differ slightly. Section 10.4.7 explains how they differ. At a DECnet–RSX/PSI node, DLM circuits to be mapped to PSI are not associated with specific physical lines. The identification of such a circuit starts with the mnemonic DLM.

For each circuit, the system manager must design various parameters, which differ depending on the type of circuit (for example, whether it is associated with a DDCMP point-to-point or multipoint line or with a PSI virtual circuit). See the appropriate system manager's guide for further information about your system's circuit parameters.

10.4.7 Multipoint Line and Circuit Parameters

A multipoint line is a single communications line connected to more than two nodes. (A line connecting two nodes is called a point-to-point line.) The DECnet implementations supporting multipoint are the Phase III and Phase IV versions of DECnet–VAX and DECnet–RSX and the Phase III versions of DECnet/E and DECnet–RT. Figure 10–1 is a diagram of a multipoint line, showing multipoint components.

The control station is the device controller responsible for overseeing data transmission to and from all the nodes attached to the line. The devices attaching the other nodes to the line are called tributaries. A DECnet–RSX, DECnet–VAX, or a DECnet/E node can support either a control station or a tributary device. A DECnet–RT node, on the other hand, can support only a tributary device on a multipoint line.



LEGEND

DV	DV11-AA/BA synchronous line multiplexer
KDP	KMC11/DUP11-DA synchronous line multiplexer
KDZ	KMC11/DZ-11-A asynchronous line multiplexer
DMP	DMP11 synchronous link
DMV	DMV11 synchronous link

Figure 10–1: A Multipoint Line

From the perspective of the control station, the multipoint line and the tributaries connected to it constitute a single line, but the separate paths to each tributary represent individual circuits. A multipoint line, therefore, has more than one circuit associated with it. (Figure 2–1 illustrates the relationship between a multipoint line and the circuits that correspond to its tributaries.)

A tributary supports only one physical link to the control station. From the tributary's perspective, the DDCMP line that links it to the control station is point-to-point. The line and corresponding circuit are therefore equivalent.

In the context of full-routing implementations, whether or not a node supports a control station or a tributary on a multipoint line is not significant. The mechanisms for handling data transmission on multipoint lines are transparent to the modules in the Routing layer of the architecture.

At the control station's node, a system manager needs to define several parameters that affect the operation of the multipoint line and its corresponding circuits:

- **Tributary addresses.** The data base at the control station's node must contain correct tributary addresses. The system manager must therefore record the unique decimal line address of each tributary on the line.
- **Polling ratios.** Whenever necessary, the control station delivers data addressed to tributaries under its control. In order to handle data originating from its tributaries, the control station periodically polls them; that is, the control station periodically asks each tributary whether it has data waiting to be transmitted. When the control station polls a tributary that has such data, it allows the tributary to transmit.

The frequency with which a tributary is polled depends on the frequency of its data transmissions. For efficiency, the control station polls active tributaries more often than inactive or dead tributaries. A dead tributary is one that has not responded within a predefined period of time.

A system manager can exercise control over the polling of specific tributaries. If for some reason a tributary should not be polled as often as others, the system manager can issue a command to assign an active polling ratio to that tributary. A command can also be issued to set a dead polling ratio that applies to all inactive tributaries.

A polling ratio is a number from 1 to 255. If a tributary's active polling ratio is 5, the control station passes through the active polling list five times before polling that particular tributary. (DECnet/E does not implement the polling technique discussed here.)

10.4.8 Transmission Mode

The system manager sets the transmission mode for every line or circuit connected to the node. The transmission mode is either half duplex or full duplex. See Section 2.2.1 for a definition of these terms.

10.5 Operating a Node

To start up a node, a system manager or operator issues commands from a terminal to load and activate required DECnet software and to turn on physical communication lines. In response to the start-up procedure, the local DECnet software initializes the DECnet software in adjacent nodes connected by the activated lines (see Section 10.4.2). Shutting down the node reverses the procedure; commands are issued to halt network activity involving the local node, to shut off physical lines, and to unload DECnet software.

10.5.1 Controlling the State of a Node

For most implementations of DECnet, a system manager turns a node on and off by manipulating the node's state; for example:

```
NCP> SET EXECUTOR STATE ON
```

This command activates the DECnet software at the node currently defined to be the executor. The executor is the node at which the NCP command actually executes. The system manager, using an NCP command, determines whether the executor is the local or a remote node.

Most DECnet implementations define three distinct node states: ON, SHUT, and OFF.

- **ON.** The local node is enabled for performing network functions.
- **SHUT.** The node maintains all existing logical links but does not permit any new links to be created. When existing links are disconnected, the node's state changes to OFF.
- **OFF.** The local node cannot participate in any network activity, and existing logical links are aborted.

10.5.2 Controlling Line or Circuit State

A node cannot actively participate in a network until one or more communication lines/circuits have been turned on. By issuing an NCP command, a system manager sets the state of a line/circuit to ON, OFF, or SERVICE:

- **ON.** The line/circuit is available for use by the DECnet software responsible for routing data packets. When a line/circuit is turned on, the local node exchanges initialization messages and, optionally, node passwords with the remote node at the other end of the line/circuit, provided that the owner of the line/circuit is NSP, the default condition (see Section 10.4.2).
- **OFF.** The line/circuit is not available for any kind of network activity.
- **SERVICE.** The line/circuit is available for special network functions only: down-line loading, up-line dumping, or loopback testing. (Note that some implementations of DECnet do not recognize SERVICE as a state explicitly separate from ON; such implementations may impose the SERVICE state on a line/circuit by their own internal means.)

When a line/circuit is in the ON state, DECnet software exercises the data link protocol that ensures data integrity and sequentiality for normal network transmissions. (The standard DECnet protocol for normal traffic is DDCMP.) In the SERVICE state, a line/circuit transmits data embedded in a protocol provided for the special network functions. The standard protocol for these functions is called the Maintenance Operation Protocol (MOP).

10.6 Monitoring Node Activity

At each node, DECnet continuously monitors the current state of the node as well as its ongoing performance. A system manager can use network management utilities to display the following information at a terminal:

- The current state of local and remote nodes and of physical lines
- Values currently defined for configuration data base and other static parameters
- The contents of various counters that DECnet software maintains to track network performance

In addition to displaying information on request, DECnet automatically logs certain events both at the operator's console and in a file. Note that remote system consoles at executor nodes can monitor operation of the Phase IV communication servers (the DECnet Router, the DECnet Router/X.25 Gateway, and the Terminal Server). Event logging records operational events such as a line starting up or shutting down. DECnet–RT does not support event logging.

DECnet uses counters to track other kinds of events and errors. A system manager can periodically record these counters in a file or display them at a terminal to obtain detailed statistics on the node's network activity. Node counters maintain statistics on logical link operations — for example, how

many connect requests have been sent and how many received and how many messages have been sent over logical links and how many received. In Phase III and Phase IV implementations, counters record Routing layer activity as well — for example, how many errors of different kinds have been found in packet headers and how many line initialization and line verification failures have occurred.

Line counters monitor events on individual communication lines. By periodically checking each line's counters, a system manager can gauge its performance and watch out for potential line problems. Line counters record statistics such as the number of data blocks sent and received successfully, the number of blocks received with errors and the number of times a tributary has passed from active to dead state.

After counters have been displayed or recorded in a file, NCP commands can be issued to set the counters to zero. In this way, a system manager can manipulate the time span that the counters monitor. For example, a system manager could set all node counters to zero as programmers begin to test a network application. At the end of the test, the counters could be examined to see how the application affected the node's performance.

Chapter 11 discusses some tools that the system manager can use in testing network components and in monitoring network activity.

10.7 Down-line Loading

DECnet-VAX, DECnet-RSX, DECnet-IAS, and DECnet-20 support down-line loading, which means loading a memory or system image from a file at one node to a separate target node. The target node is usually an RSX-11S DECnet node or a communication server node (DECnet Router, DECnet Router/X.25 Gateway, or Terminal Server. These are memory-only systems with no disk-based file storage of their own.

NOTE

A DECnet-20 node loads a system image down-line to a specially adapted RSX-11S node called a DN200, which supports a card reader and a line printer and which serves as a remote batch station. The DN20, a communications front-end, is loaded in the same way.

At a DECnet-VAX, DECnet-RSX-11M or M-PLUS node, the system manager generates the target system image. Once generated, the image can be modified by VNP commands at an executor node. The load itself can be initiated in one of two ways. An operator can issue NCP commands to load the image down-line to the target node, or an operator at the target can initiate the load by triggering a bootstrap ROM (read-only memory). (Section 10.7.3 explains these procedures.)

Up-line dumping is a function that complements down-line loading. This capability is described in Section 11.1.3.

10.7.1 Down-line Loading Definitions

The down-line loading function is distributed among two or more nodes in a DECnet configuration. The following definitions clarify the roles played by the various nodes.

- **The command node** is the node from which the NCP load commands are issued.
- **The executor node** actually executes the NCP commands; it must be adjacent to the target node.
- **The target node** receives the system image loaded down the line or dumps a system image up the line (see Section 11.1.3).

A single node can act as both the command and the executor node.

10.7.2 Down-line Loading Data Base Parameters

For every target node to be down-line loaded, the executor has access to a permanent data base. Each data base contains default parameters, such as node verification passwords and other account information, for down-line loading a specific target node. The system manager can override these defaults by providing parameter values in an NCP LOAD command. The parameters are defined initially at network generation and can be redefined when necessary.

10.7.3 Performing a Down-line Load

Whether a remote command node or the target itself initiates the load, the target must have local access to a cooperating program called a primary loader. This loader is usually contained in a bootstrap ROM (read-only memory) incorporated in the target. During the down-line load procedure, a series of programs can be loaded on top of the primary loader. Each program calls the next until the system image itself is loaded down-line.

The line or circuit between the executor and the target is in a SERVICE state during the procedure (see Section 10.5.2). Either the system manager explicitly sets the state to SERVICE or the DECnet software sets the state automatically. How the state is set depends on the implementation and on the way a load is initiated.

10.7.3.1 The LOAD Command — The NCP LOAD command is the means of initiating a down-line load from a remote command node. As soon as the LOAD command has been issued, an operator at the target must manually trigger the bootstrap ROM unless the service line's device controller is a DMC11 or DMR11 device. These devices can trigger the target's primary loader automatically if the LOAD command passes down the correct password (see Section 10.7.2). Figure 10–2 illustrates a down-line load initiated by a command node.

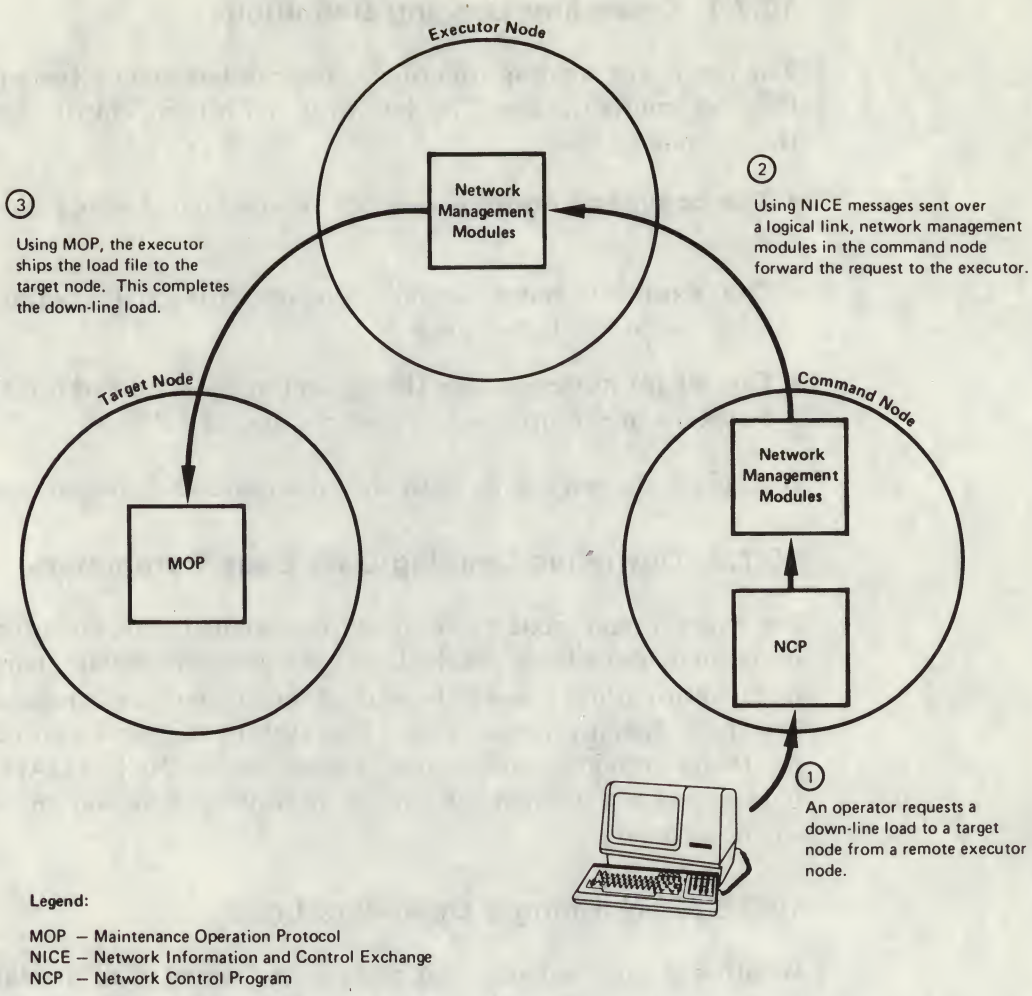


Figure 10–2: Down-line Load Initiated by a Command Node

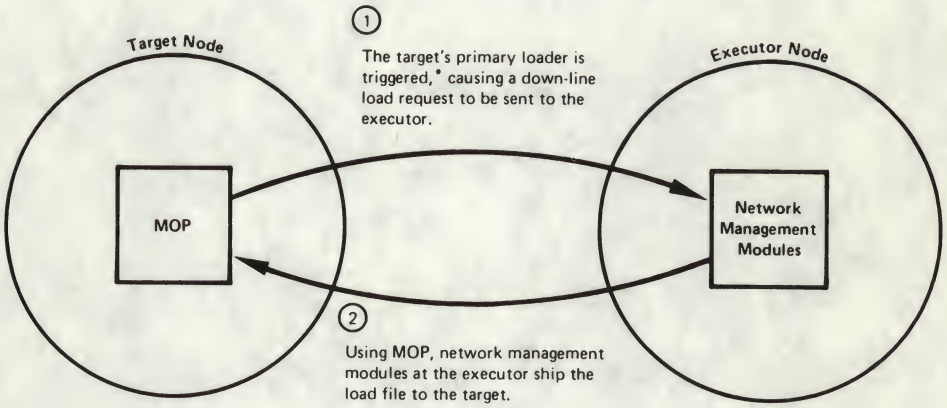
10.7.3.2 Target-initiated Down-line Loads — An operator at the target node can request a load by manually triggering the primary loader. In addition, the loader is triggered automatically at the completion of an up-line dump from the target. Target-initiated down-line loads always use the parameter values defined in the permanent data base for the target. Figure 10–3 illustrates a target-initiated down-line load.

10.7.4 Down-line Loading and Checkpointing RSX–11S Tasks

DECnet–11M, DECnet–11M–PLUS, and DECnet–VAX support two capabilities relating to a DECnet–11S node. The first is called down-line task loading. RSX–11S tasks can be stored at a DECnet–11M, DECnet–11M–PLUS, or DECnet–VAX node and loaded to the RSX–11S node. The second is called checkpointing, which is a standard RSX–11M capability. An executing RSX–11S task can be interrupted, copied in its interrupted state up the line, and then be replaced by a higher priority task loaded down-line from the executor. When the higher priority task has completed, the interrupted task is reloaded down-line and allowed to continue executing.

At a DECnet–11M or DECnet–11M–PLUS node, the operating system regularly checkpoints tasks to local disk storage. However, RSX–11S nodes are basically memory-only systems, so the only way to checkpoint tasks is to use the executor's disk storage.

These two capabilities give flexibility to an RSX–11S node that would not be possible without DECnet. To change the set of resident tasks at a stand-alone RSX–11S system, an operator would have to reboot with a different system image. See the *DECnet–RSX System Manager's Guide* and the *DECnet–VAX System Manager's Guide* for further information.



*An operator triggers the loader manually or the completion of an up-line dump triggers it automatically.

Legend:

MOP – Maintenance Operation Protocol

Figure 10–3: Down-line Load Initiated by a Target Node



Chapter 11
Monitoring and Testing DECnet Performance

Computer networking creates a highly dynamic environment. Applications and other processes are exchanging information simultaneously over thousands of circuits; functions at various levels are being enabled/disabled; the status of network components is changing rapidly; internetwork communications can be making multiple demands for protocol handling; warning conditions can arise and fatal events occur in different regions of the network and at various levels of network function. And as the number of nodes increases, so does the volume and complexity of activity.

Maintaining a high level of network availability under such conditions is a challenge to network management. The network manager must have tools that report on, diagnose, and correct conditions that could cause performance degradation or failures. Facilities must also be available for monitoring network operation so as to detect conditions that, if left to develop, could lead to major communications problems.

Several software tools, both integrated into DECnet and available separately, are designed to help the network manager in these assignments. The integrated testing tools perform the following functions:

- Test that user-specified nodes are communicating properly and that messages move over specified circuits and lines without being corrupted
- Test operation of X.25 and SNA Gateways to determine whether or not protocols are being handled properly
- Trigger memory dumps to help you establish the cause of node crashes

The separately available monitoring tool is a product called Observer. Running in a DECnet–RSX node, this product alerts you to incipient network problems and also gathers performance statistics that can help you to reconfigure and fine tune the network for optimal performance. Each of these tools is discussed in the following sections.

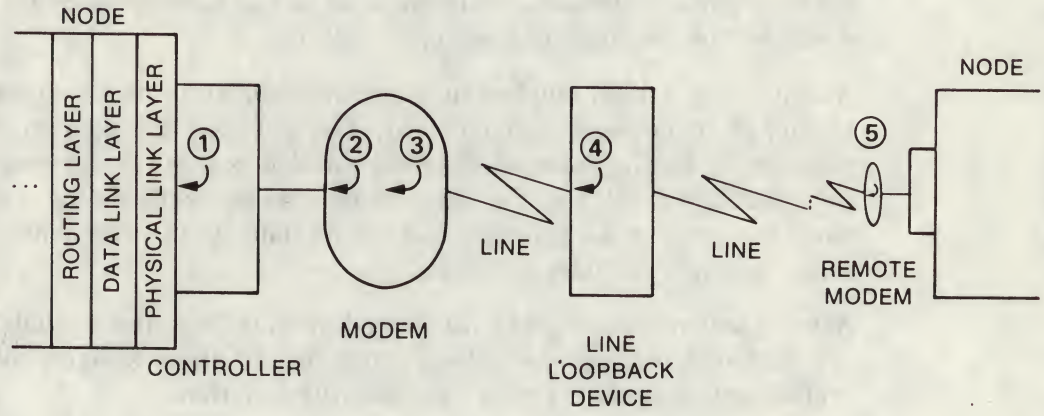
11.1 Integrated Testing Tools

The following tools are integrated components of most DECnet implementations. In other words, when you install DECnet, these testing capabilities are available.

- Loopback testing
- Trace capability
- Up-line dump

11.1.1 Loopback Testing

Loopback tests exercise software and hardware by repeatedly sending data through a number of network components and then returning the data to its source. Figure 11–1 illustrates the hardware points from which test data may be looped back to the point of origin. If a test completes successfully, the data loops back to its source uncorrupted; if a test fails, the data either does not return to its source or returns in a corrupted state.



LEGEND

① Loopback at controller

② Loopback at modem on controller side

③ Loopback at modem on line side

④ Loopback at hardware loopback device inserted in line

⑤ Loopback at a remote modem

Figure 11–1: Hardware Loopback Devices

A system manager can run variations of the loopback tests to help isolate network components responsible for losing or corrupting data. Digital software services personnel routinely run loopback tests after installing DECnet software. Successful tests verify that both the software modules and the hardware equipment of the path tested are operating correctly.

The loopback-testing capability is exercised through the NCP LOOPBACK command in combination with other NCP commands. Loopback enables you to check whether or not logical links are properly established between nodes, whether or not DECnet protocols operate properly within and between nodes, and whether or not messages are being transmitted properly over the circuits and lines that you specify.

Figures 11–2 and 11–3 show the NCP commands used in conjunction with command- and program-initiated node-level loopback tests, respectively. Figure 11–4 shows the NCP commands used in conjunction with line and circuit level loopback testing.

Test data is transmitted from a source node to a local modem, a point on a line connecting two nodes, a remote modem, or to a remote node and looped back to the source node. Test results that are printed out indicate whether or not the target network component that you specified (line, circuit, or node) tested out successfully. Test messages specify the nature of an error condition that may have caused the test to complete unsuccessfully.

Loopback testing can be performed for general purpose nodes and communications servers over Ethernet, DDCMP, SNA, or X.25 circuits.

Used in combination, loopback testing of both nodes and circuits can help you isolate problem areas in the network. Procedures to be observed for each of the loopback tests are specified in the documentation of each DECnet product.

11.1.2 Trace Capability

The trace capability is useful in defining protocol problems that may arise in the gateway products that handle internetwork communication, and in the PSI (Packetnet System Interface) products that enable nodes and non-DECnet Digital systems to communicate over PPSNs. For the DECnet Router/X.25 Gateway, the trace capability is exercised by means of the NCP TRACE command; for the DECnet/SNA Gateway, the trace capability is exercised by means of the SNATRACE utility. These capabilities enable you to capture and analyze those portions of messages that define how protocols are being handled.

Protocol processing by the DECnet Router/X.25 Gateway can be TRACEed at levels 2 and 3 of the protocol-handling software. Level 2 defines the link access procedure for data exchange between computers and the data communications equipment that establishes, maintains, and terminates communications between computers over a PSN. The level 3 protocol defines the procedures for formatting and exchanging packets transmitted over a PSN.

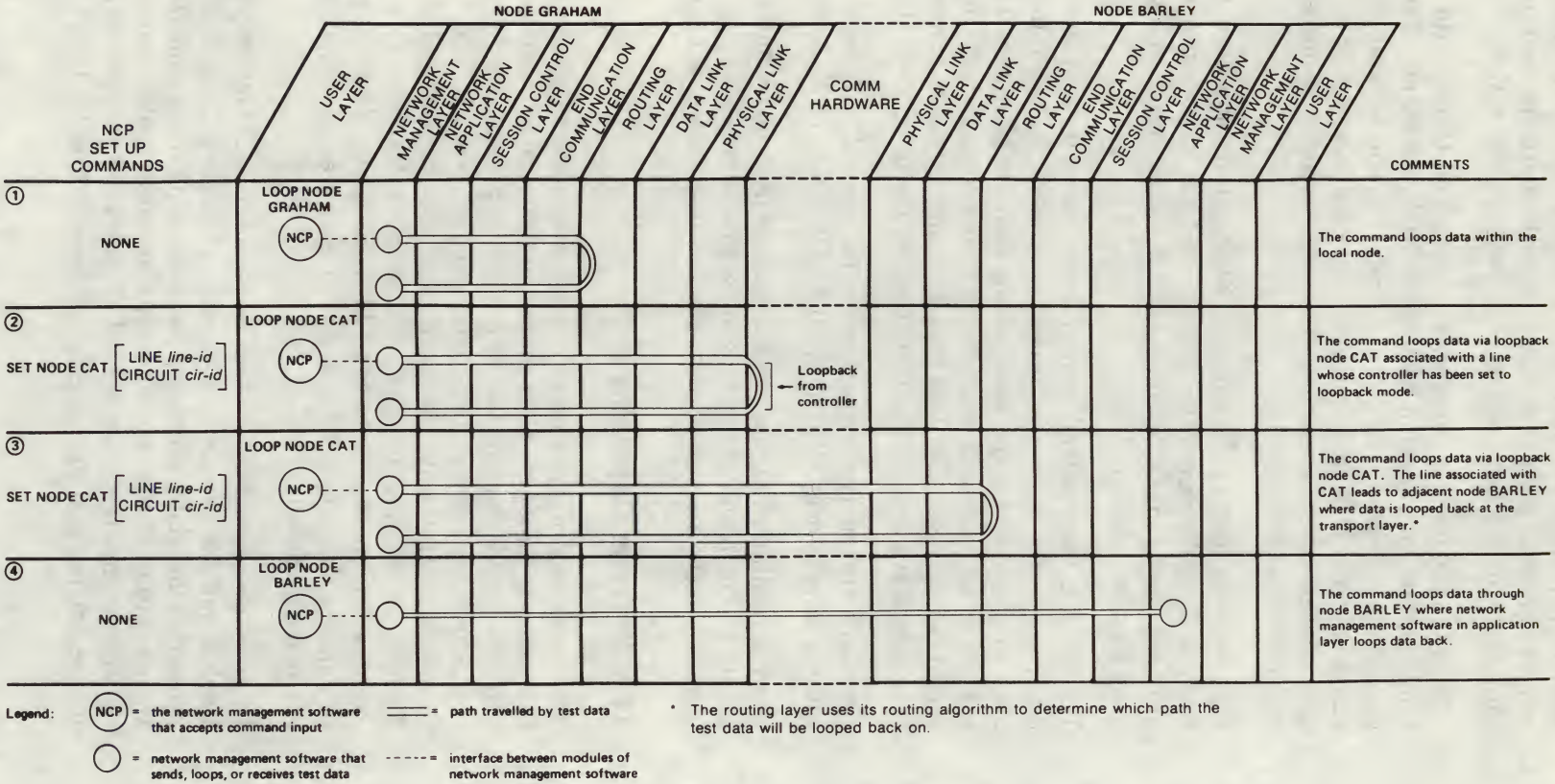


Figure 11–2: Command-initiated Loopback Tests

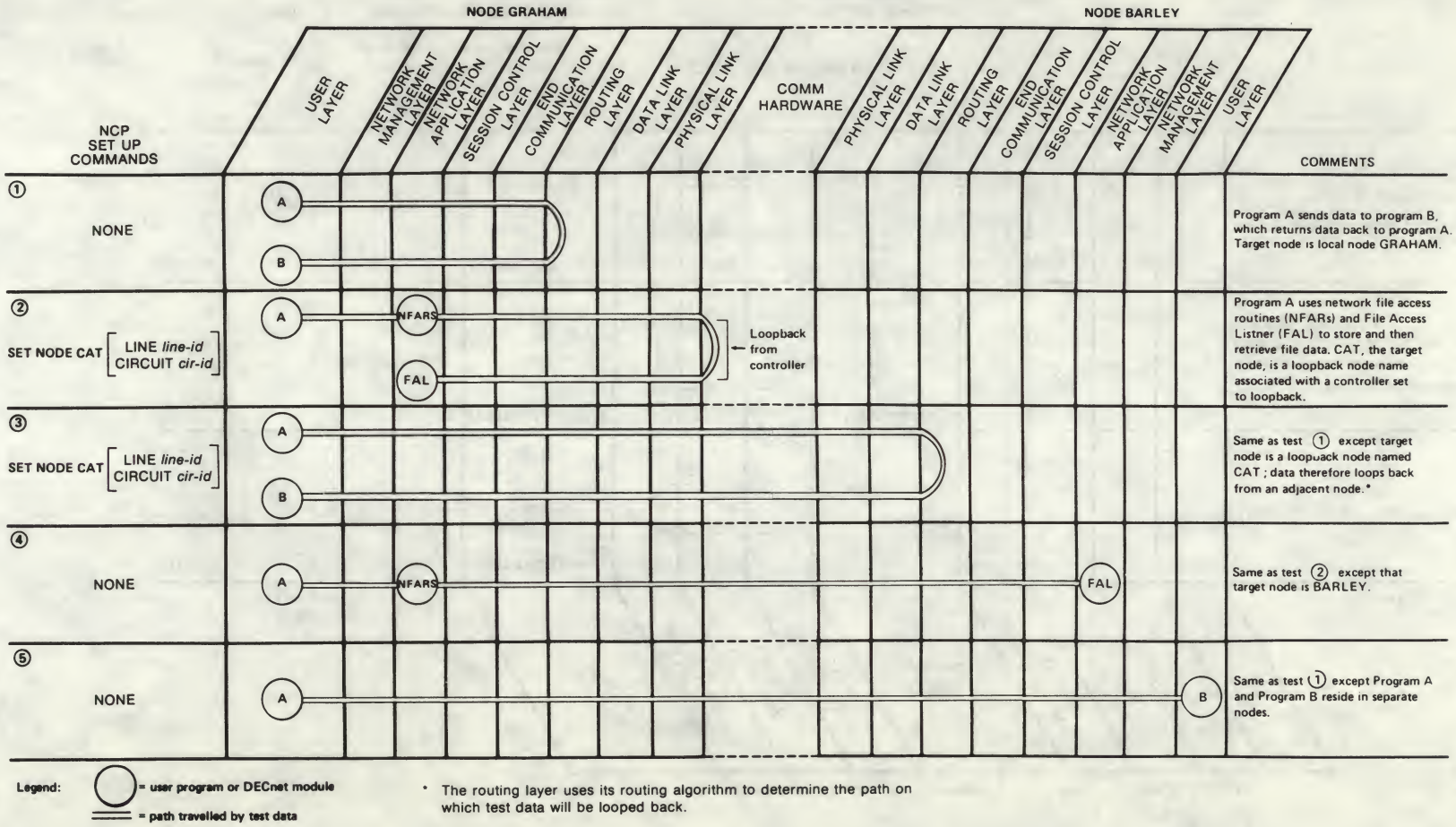


Figure 11–3: Program-Initiated Loopback Tests

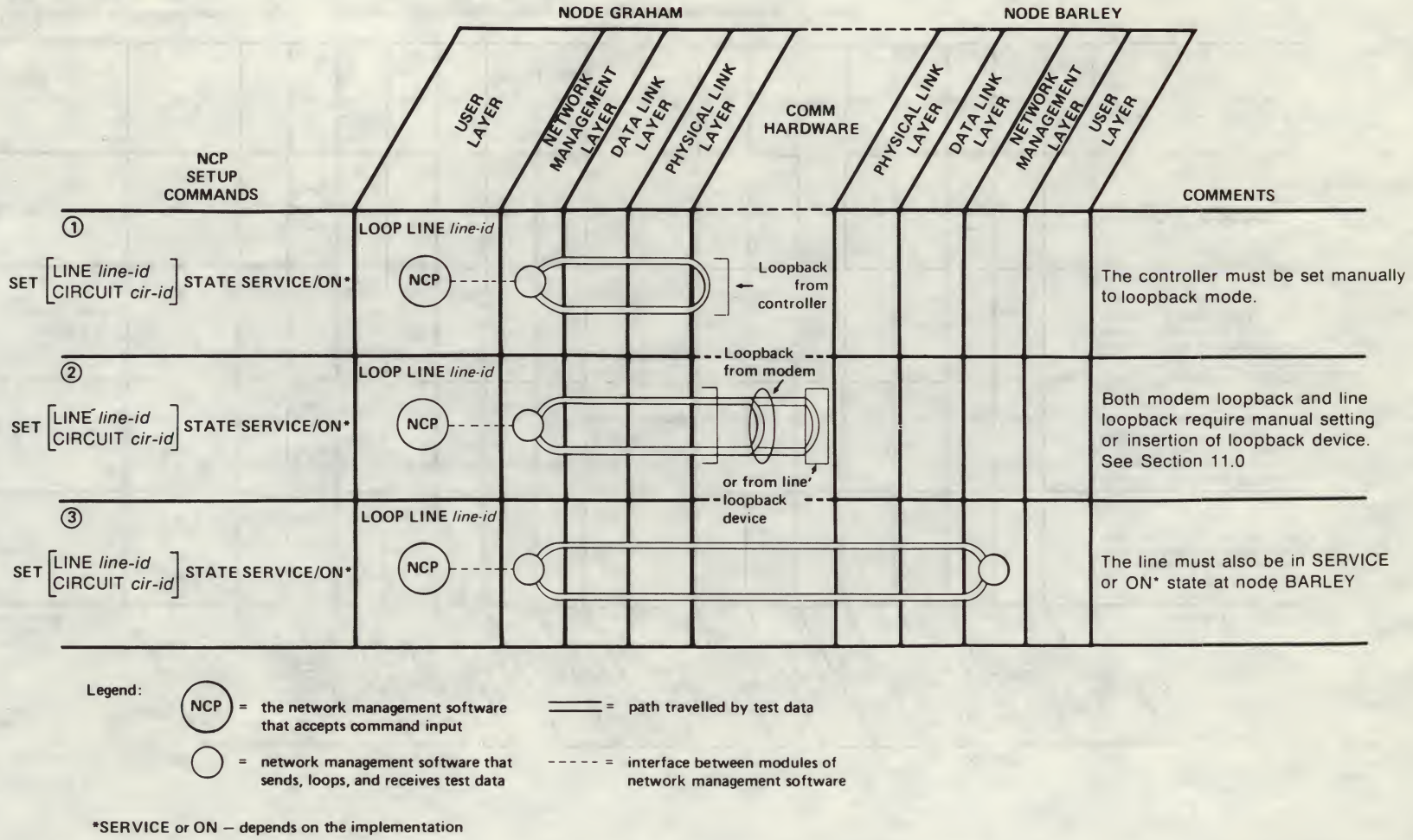


Figure 11–4: Line/Circuit Level Loopback Tests

SNATRACE, used for identifying protocol problems that can arise in using the DECnet/SNA Gateway, enables you to examine protocol handling at the SDLC circuit level, the physical unit level, and the session level. At the SDLC circuit level, you retrieve messages on a specified circuit as they pass between the SDLC module and the device driver in the Gateway node. At the physical unit level, you retrieve messages handled by a specified physical unit as they pass between the SNA module and the SDLC module of the Gateway. At the session level, you retrieve messages being exchanged within a specified session as they pass between the SNA and SDLC modules of the Gateway.

Documentation for each of the Gateway products tells you how to use these facilities.

11.1.3 Up-line Dump

Certain internode relationships in DECnet are characterized by the terms "executor" and "target." An executor node is one that executes NCP commands that affect adjacent target nodes. Insofar as discussion of the up-line dump capability is concerned, a target node is either a DECnet RSX–11S node, or a communications server node. The executor node in both these cases is usually the one from which the target node's system was down-line loaded, or a backup node of similar capability in the event that the original executor is not available.

The up-line dump capability enables a target node to automatically initiate a memory dump up-line to the executor node. This will occur when the target node senses an impending system crash. By analyzing the memory dump, software specialists can determine the cause of the system failure and take steps to assure that similar conditions do not arise again.

In order to keep the target node available to the network, the executor node will automatically down-line load the target's system upon completion of the dump. Detailed descriptions of the DECnet facilities that operate in up-line dumping and of the NCP commands to be used to assure that lines and circuits specified in the procedure can accommodate the dump are presented in the documentation of the systems in which the capability resides.

An up-line dump can also be triggered by the system manager at the executor node at any time, or can be done automatically if the remote system crashes.

11.2 Monitoring Network Operation with Observer

Observer is a software product that monitors the performance of DECnet networks. Installed in a Phase III or Phase IV DECnet–RSX node, it gathers, processes, and generates information on network performance. Network events such as changes in line, link, and node states are displayed as they occur on a VT100 terminal dedicated to this function; other information is logged to files that can be processed periodically to produce network performance reports. A command terminal that offers you a menu-based approach to network surveillance enables you to display a broad range of information concerning the state of designated network components. You can, for example, retrieve statistical information regarding network operation and also enter commands affecting the reporting relationship between Observer and specified network components or regions.

The information that appears on the event display terminal can alert you to conditions that might cause problems in network operation. Forewarned, you can then take steps to correct such conditions before major communications or performance problems develop. If, for example, a message from Observer notifies you of high data traffic over a specific link, it might mean that certain nodes are processing and transmitting unexpected data loads or that you originally specified too low a data rate for that link. Proper action, through Observer facilities, in respect to the nodes and the link could then improve performance in that part of the network.

By means of a menu-based approach as shown in Figures 11–5, 11–6, and 11–7, the command terminal enables you to monitor more closely, specific regions and components of the network and to control Observer operation.

Figures 11–5, 11–6, and 11–7 illustrate just one example of how you can use Observer to focus in on various network areas and components. Figure 11–5 illustrates the Master Menu that appears on the command terminal. If you select "A" from this menu, for example, the Network Summary Status Display (Figure 11–6) appears. This screen, which is updated about every 30 seconds, displays summary information concerning the network being monitored. In addition to presenting identification information, it specifies the number of nodes reachable and unreachable from the Observer node, the results Observer obtained, and other statistics gleaned in the most recent polling cycles. The names of the nodes in the network or network regions are also displayed. Network regions are user-defined. Analysis of the information logged by Observer can serve as a basis for reconfiguring the network and tuning it for optimum performance.

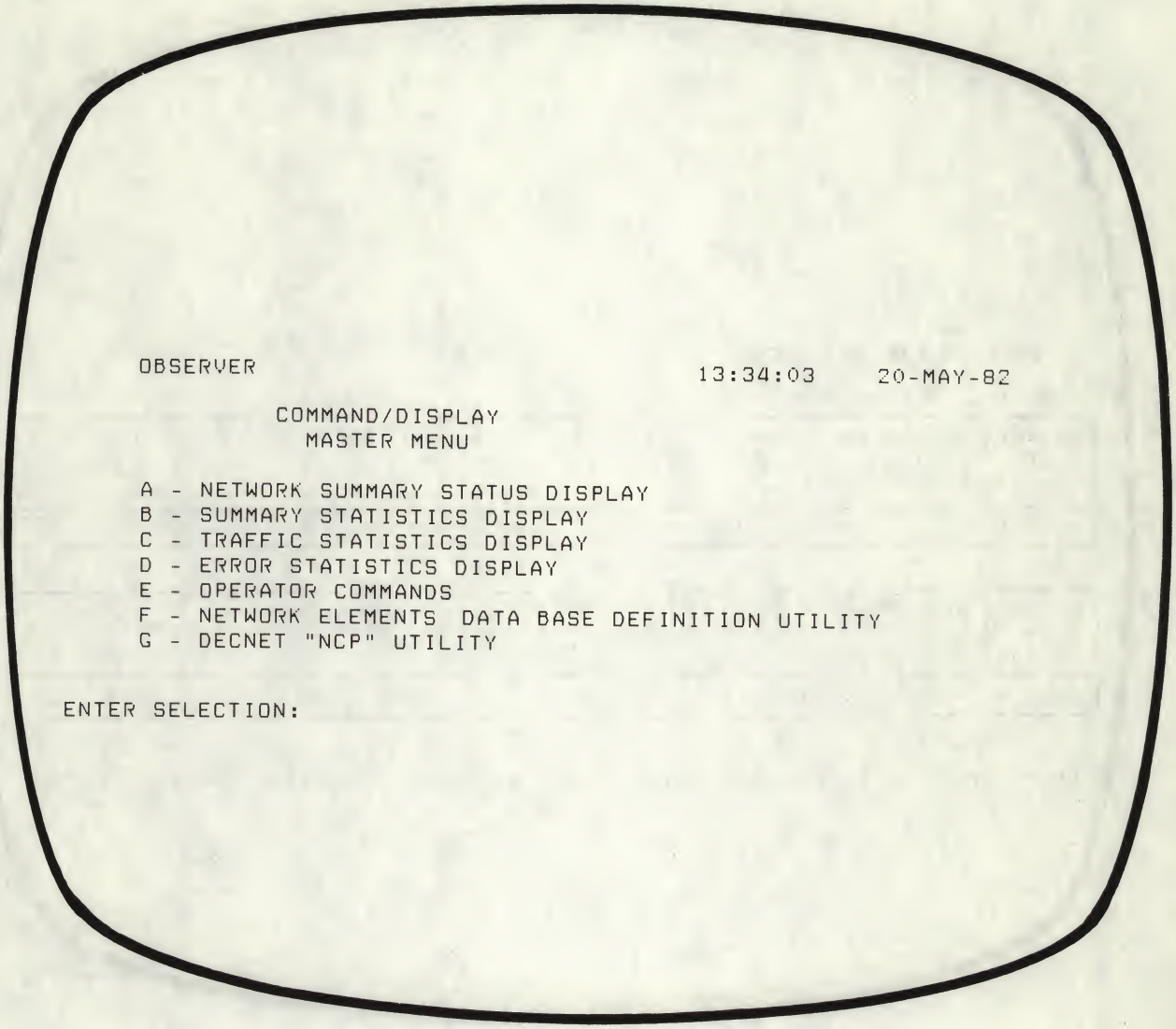


Figure 11–5: Observer Master Menu

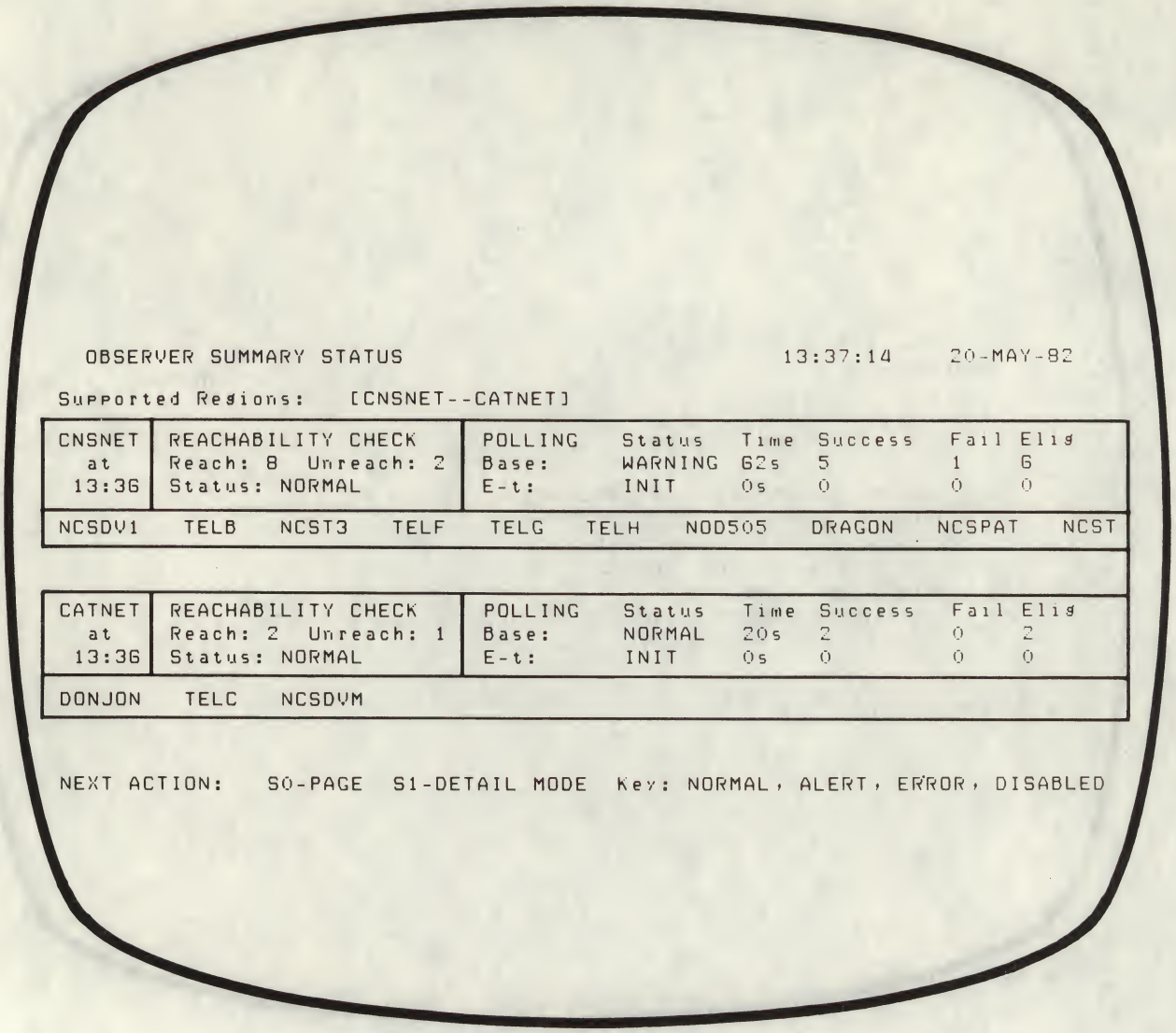


Figure 11–6: Observer Summary Status Display

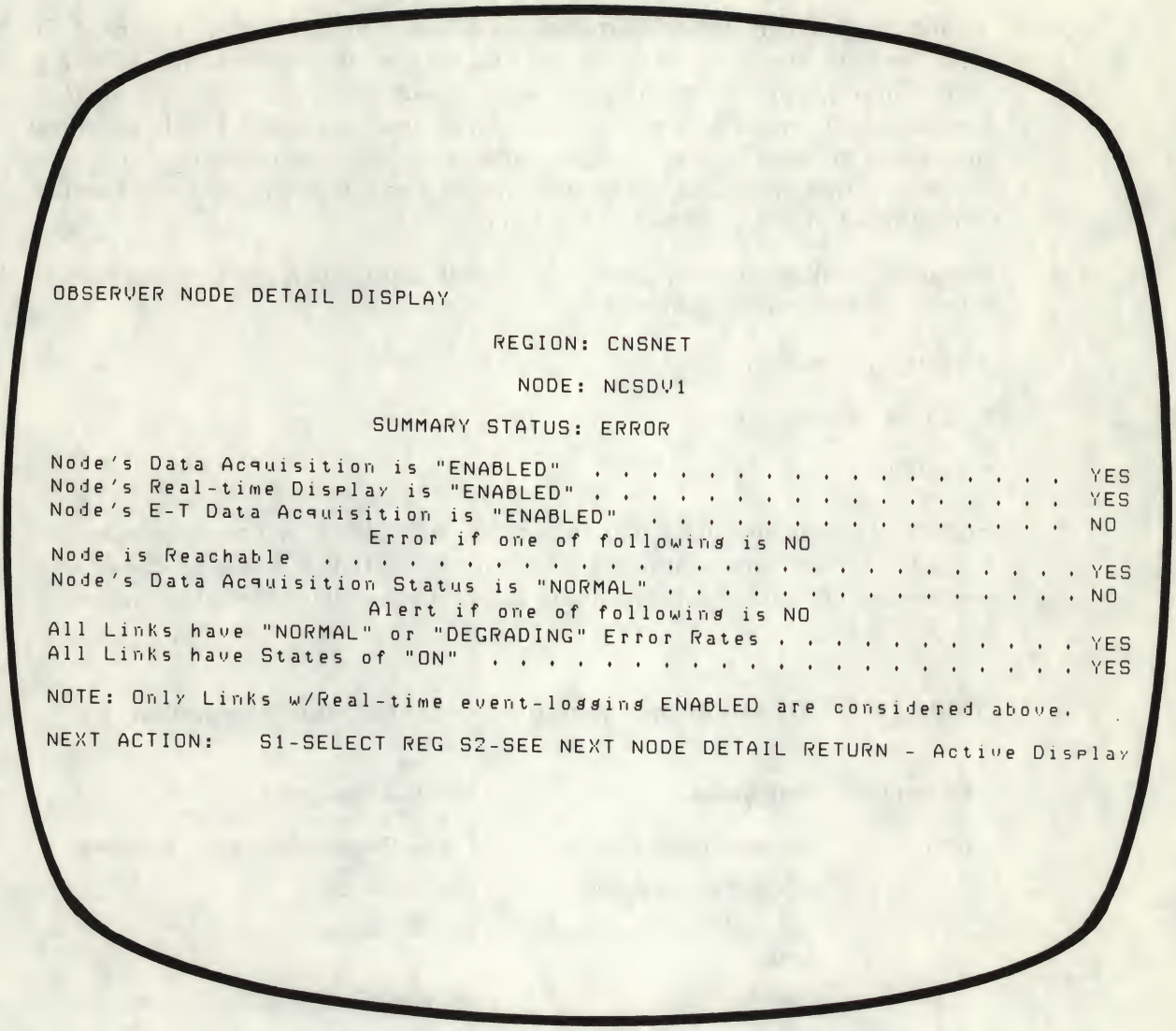


Figure 11–7: Observer Node Detail Display

To focus in further, you can summon up a Node Detail Display (Figure 11–7) for a particular node. As shown in the illustration, this screen displays certain status information concerning the node selected. If you need still further information in order to diagnose a problem, you can select another display directly or go back to the Summary Status display. Detailed information on the ease with which Observer enables you to move from one screen to another is contained in the Observer User's Guide.

Observer monitors nodes, links, end-to-end traffic, and network regions. It reports three kinds of information:

- State information
- Status information
- Statistics

State and status information are identified in Table 11–1. The kinds of statistics gathered and maintained by Observer are listed in Table 11–2. See Observer documentation for details on function, installation, and operation of the product.

Table 11–1: Observer-maintained State and Status Information

Element	State/Status	Possible Conditions
Link	Link error rate status	Normal/Degrading/Degraded/Unusable
	Link traffic rate status	Normal/High/Low
	Link real-time event-logging status	Enabled/Disabled
	Link state	Initialization/On Line/Unknown/Removed
Node	Base system and end traffic enable status	Enabled/Disabled
	Node data acquisition status	Unreachable/Normal/Unknown/Error
	Node traffic rate status	Normal/High/Low
	Node real-time event-logging status	Enabled/Disabled
	Node state	Initialization/Normal/Warning/Fatal Error
Region	Base system and end traffic enable status	Enabled/Disabled
	Node reachability check completion status	Initialization/Normal/Warning/Fatal Error
	Base system and end traffic polling cycle completion status	Initialization/Normal/Warning/Fatal Error

Table 11–2: Statistics Maintained by Observer

Type of statistic	Element	Statistics
Time related	Node	Time base in seconds
		Time last data received
		Time of last state
	Link	Time base in seconds
		Time last data received
		Time of last state change to on
Error related	Node	Buffer resource errors
		Packet format errors
		Partial routing update loss
		Aged packet loss
		Node unreachable packet loss
		Out-of-range packet loss
		Oversize packet loss
		Network errors
		Node errors
	Link	Data blocks sent
		Data blocks received
		Collision detect check
		Failure (Ethernet only)
		Data errors outbound
		Local reply timeouts
		Data errors inbound
		Local buffer errors
		Unknown frame destination (Ethernet only)
		Congestion loss
		Selection timeouts
		Resets
		Link error level
		Link down
Traffic related	Node	Bytes received
		Bytes sent
		Node traffic level (bytes)
		Packets received
		Packets sent
		Node traffic level (packets)
	Link	Link traffic level (bytes)
		Link traffic level (packets)
		Bytes sent
		Data blocks sent
		Data blocks received
		Transit packets sent
		Originating packets sent
		Blocks deferred (Ethernet only)
		Blocks sent with collision (Ethernet only)
		Bytes received
		Transit packets received
		Terminating packets received
		Multicast blocks received (Ethernet only)

INDEX

A

Aborting a logical link, 5–8
Access control, 4–6, 10–6
 connect block information, 5–3
 for remote file access
 terminal operations, 6–6
 for remote file access programs, 6–4
Access methods
 for X.25, 8–4
Address
 node, 2–1
 source, 2–1
ALL-IN-1, 9–7
Applications, 9–1
 accounting/financial subsystem, 9–1
 ALL-IN-1, 9–7, 9–15
 billing and delivery scheduling, 9–2
 component inventory, 9–2
 controlling a logical link, 9–6
 COPY command, 9–7
 DECnet/SNA gateways, 9–13
 destruction of a logical link, 9–6
 establishing a logical link, 9–5
 file transfer, 9–6
 logical links, 9–4
 manufacturing subsystem, 9–1
 marketing subsystem, 9–1
 network virtual terminals, 9–11
 order entry, 9–2, 9–4
 packet-switching networks, 9–3
 Phone, 9–14
Applications (Cont.)
 production scheduling, 9–2, 9–4
 raw materials inventory, 9–2
 remote file access, 9–11
 SET HOST command, 9–11
 task-to-task communication, 9–3
 terminal facilities, 9–13
 TLK, 9–14
 X.25, 9–13

B

Buffer size, 10–7
Bus topology, 3–13
 see also Ethernet

C

Carrier Sense Multiple Access with
 Collision Detection, 3–13
CFE, 10–3
Checkpointing RSX–11S tasks, 10–14
Circuit cost, 2–7
 see also Path cost
Circuit states
 how to control them, 10–11
Circuits, 1–1
 configuration factors, 3–12
 counters, 10–11
 DECnet/SNA, 8–9
 definition, 2–3
 modifying parameters, 10–8

Circuits (Cont.)
multipoint connections, 3–12
multipoint parameters, 10–8
point-to-point connections, 3–12
SDLC, 8–9
X.25, 8–3
Communicating with foreign vendors, 1–9
see also Gateway access
Communications servers, 3–11
DECnet/SNA Gateway, 3–12
Router server, 3–11
Router/X.25 Gateway, 3–12
terminal server, 3–11
Configuration, 3–1, 3–13
data bases, 10–3
environments, 3–2
line and circuit characteristics, 3–1, 3–12
local area networks, 3–4
major factors, 3–1
node characteristics, 3–1, 3–8
wide area networks, 3–2
Configuration File Editor, see CFE
Connect block, 5–2
see also Connect request
parameters, 5–3
Connect request, 5–2
see also Logical links
CSMA/CD, 3–13

D

Data bases
configuration, 10–3
distributed, 1–9
Data link independence, 1–4
Data link mapping (DLM), 8–4
Data links
layers, 1–6
protocols, 2–5
Data types, 4–5, 5–6
for remote file access, 6–5
DCE, 8–2
DDCMP, 1–4
Decentralization, 1–1
DECnet
accepting/rejecting logical link requests, 5–6
applications, 9–1
capabilities, 1–1, 4–1
Phase III, 1–7
Phase IV, 1–8
concepts, 2–1
configurations, 3–1
DECnet (Cont.)
definition, 1–1
development phases, 1–7
distributed data bases, 1–9
environments, 3–2
flow control, 4–6, 5–7
link service messages, 4–6
local area networks, 3–4
logical link creation, 4–2
network management, 1–3, 10–1
performance, 11–1
resource sharing, 1–9
sending and receiving data, 5–6
task-to-task calls, 5–1
terminal facilities, 7–1
testing tools, 11–1
topology, 1–3
wide area networks, 3–2
DECnet phases
configuration factors, 3–8
DECnet Router/X.25 Gateway, 8–5
trace facility, 11–3
DECnet/SNA
capabilities, 8–9
circuits, 8–9
communications, 8–7
environments, 8–8
loading software, 8–9
DECnet/SNA Gateway, 3–12, 8–1, 8–7
implementations, 3–12, 8–9
trace facility, 11–3
Dedicated nodes, 3–10
Defining configuration parameters, 10–5
Defining network parameters, 10–1
see also Network management
DELNI, 3–6
see also Ethernet
DEQNA, 3–5
see also Ethernet
DEUNA, 3–5
see also Ethernet
Digital Data Communications Protocol (DDCMP), 2–5
Digital Network Architecture
protocols, 1–6
Digital Network Architecture (DNA), 1–4
interfaces, 1–5
layers, 1–5
DLM, see Data link mapping
DNA dump/load protocol, 2–4
see also Ethernet circuits
DNA remote console protocol, 2–4
see also Ethernet circuits

DNA routing protocol, 2-4
see also Ethernet circuits
DNA, see Digital Network Architecture
Down-line loading, 1-3, 10-2, 10-12
see also Network management
checkpointing RSX-11S tasks, 10-14
data base parameters, 10-13
how to do one, 10-13
target initiated, 10-14
terminology, 10-13
DTE, 8-2

E

End Communication layer, 1-6
End node, 3-9
see also Routing
Ethernet
circuits, 2-4
configurations, 1-3
data link independence, 1-4
definition, 3-4
hardware, 3-5
network adaptors, 3-5
protocols, 2-4, 2-5
supported node types, 3-7
the specification, 3-4
Event logging, 10-11
Executor node, 11-7

F

Fiber optic link, 3-6
Fictional Corporation, 9-1
see also Applications
File access listener, 6-2
File characteristics, 6-4
access method, 6-5
data types for remote file access, 6-5
file organization, 6-5
record attributes, 6-5
record format, 6-5
File protection, 6-6
File specifications, 6-4
for remote files, 6-7
File transfer, see also Remote file access, 1-9
Flow control, 4-6, 5-7
buffer space, 4-6
Full duplex, 2-4
see also Half duplex
Full-function node, 3-9, 3-10
see also Routing

G

Gateway access, 1-4
see also DECnet/SNA and X.25
General purpose nodes, 3-10, 3-12

H

H4000 transceiver, 3-5
see also Ethernet
Half duplex, 2-4
see also Full duplex
Handshake procedure, 4-2, 6-3
see also Logical links
Hops, 2-6
see also Routing
modifying the parameter, 10-7

I

IBM SNA network, 8-1
Indexed file organization, 6-5
Initialization messages, 10-6
Integrated testing tools, 11-2
Interfaces, 1-5
International Standards Organization (ISO), 1-4
Internetwork communications, 8-5
Interrupt data, 4-5, 5-7
ISO, see International Standards Organization

L

Line states
how to control them, 10-11
ON, OFF, SERVICE, 10-11
Lines, 2-3
configuration factors, 3-12
counters, 10-11
line identification, 10-7
multipoint, 3-12
multipoint parameters, 10-8
point-to-point, 3-12
polling, 3-13
transmission modes, 2-4
Link addresses, 4-4
see also Logical links
Link identifier, 4-4, 5-3
see also Connect block
see also Logical links
Link service messages, 4-6, 5-7
Load command, 10-13
Local area networks, 3-4

Logical link identifier, 5-6
Logical links, 4-1
 aborting a link, 5-8
 accepting/rejecting requests, 5-6
 creation, 4-2
 identification, 4-4
 requesting a link, 5-2
 sample application, 9-4
 terminating a link, 5-7
 use, 4-2
Loopback test, 11-2
 hardware devices, 11-2

M

MAIL command, 9-15
Management utilities, 10-2
 CFE, 10-3
 NCP, 10-2
 VNP, 10-3
Manchester physical channel encoding, 2-5
Modifying routing parameters, 10-7
Monitoring and controlling the network
 node activity, 10-2, 10-11
 Observer, 11-7, 11-12
 testing, 11-1
Multipoint lines, 1-3

N

NCP, 10-2
Network Application layer, 1-5
Network command terminals, 7-2
Network Control Program
 see NCP
Network File Transfer (NFT), 6-2, 6-5
Network generation, 10-1
 see also Network management
Network Generation Planning, 10-4
Network management, 1-3, 10-1
 down-line loading, 1-3
 functions, 10-1
 problem isolation, 1-4
 utilities, 10-2
Network Management layer, 1-5
Network object parameters, 10-6
 see also Object types
Network specifications, 5-5
 see also Connect block
Network virtual terminal (NVT), 7-3
 sample application, 9-11
NFT, see Network File Transfer
Node
 address, 2-1, 10-6
 configuration characteristics, 3-8
 counters, 10-11
 DECnet phases, 3-8
 dedicated, 3-10
 definition, 1-1, 2-1
 executor, 11-7
 general purpose, 3-10, 3-12
 generation, 10-2, 10-5
 names, 2-3
 target node, 11-7
Node names, 2-3, 10-6
Node operation, 10-2, 10-10
 see also Network management
Node states
 how to control them, 10-10
 ON, SHUT, OFF, 10-10
Node verification passwords, 10-6
Normal data, 4-5, 5-6
NVT, see Network virtual terminal

O

Object name, 5-3
 see also Connect block
Object type, 5-3
 see also Connect block
 reserved, 5-4
 unreserved, 5-4
Observer, 4-6, 11-1
 master menu, 11-8
 node detail display, 11-12
 operations, 11-7
 state and status information, 11-12
 summary status display, 11-8
 table statistics, 11-13

P

Packet assembly/disassembly facility (PAD), 8-6
 see also X.25
Packet-switching data network (PSDN), 1-4, 8-1
 definition, 8-2
Packetnet System Interface (PSI), 8-1, 8-5
 trace facility, 11-3
PAD
 see Packet assembly/disassembly facility
Parameters
 routing, 10-7

Passwords, 10-6
Path cost, 2-7
 see also Routing
 modifying the parameter, 10-7
Path length, 2-6
 see also Routing
Permanent virtual circuits, 8-4
 see also Circuits
Phone, 7-1
 sample application, 9-14
Physical channel encoding, 2-5
 see also Ethernet
Physical Link layer, 1-6
Physical links, 1-1
Polling, 3-13
 see also Lines
 ratios, 10-9
PRO/DECnet, 3-8, 5-1
 sample application, 9-16
Problem isolation, 1-4
Protection code, 6-4, 6-6
Protocols
 byte-oriented, 2-5
 data link, 2-5
 Digital Data Communications Protocol (DDCMP), 2-5
 DNA, 1-6
 user-created, 1-7
PSDN, see Packet-switching data network
PSI
 see Packetnet System Interface

R

Random file access, 6-5
Receive password, 10-6
Record access, 6-9
Record attributes, 6-5
Record format, 6-5
Relative file organization, 6-5
Remote command file submission, 6-7
Remote file access, 1-9, 6-9
 access control, 4-6, 6-6
 command file submission, 6-7
 file specification, 6-4
 file systems, 6-3
 file transfer application, 9-6
 functions, 6-2, 6-6
 initiating access, 6-3
 open call, 6-4
 programming applications, 6-1, 6-2
 record access, 1-3
 sample application, 9-11
Remote file access (Cont.)
 terminal operations, 6-1, 6-5
Remote job entry (RJE), 3-12, 8-9
 see also DECnet/SNA
Repeater, 3-6
 local, 3-6
 remote, 3-6
Repeater, see also Ethernet
Resource sharing, 1-9
Retransmission of data, 4-5
Router server, 3-11
Router X.25 Gateway, 3-12
Routing, 1-3
 buffer parameters, 10-7
 configuration factors, 3-9
 definition, 2-5
 end node, 3-9
 Ethernet messages, 2-5
 full-function node, 3-9
 path determination, 2-5
 specifying parameters, 2-8, 10-7
 terminology, 2-6
 timing parameter, 10-7
 X.25 messages, 2-6
Routing layer, 1-6

S

SDLC, 8-9
Segment acknowledgment, 4-5
Sending and receiving data, 5-6
Sequential file access, 6-5
Sequential file organization., 6-5
Session Control layer, 1-5
SET EXECUTOR STATE command, 10-10
Source address, 2-1
Source program, 4-2, 6-2
 see also Logical links
State and status information, 11-12
Switched virtual circuits, 8-4
 see also Circuits
System management, see Network management

T

Target node
 for up-line dumps, 11-7
Target node identifier, 5-3
 see also Connect block
 see also Node address
Target program, 4-2, 6-2
 see also Logical links

Task-to-task communication, 1-3, 5-1
 access control, 4-6
 DECnet calls, 5-1
 for DECnet/SNA, 3-12
 functions, 5-2
 sample application, 9-3
Terminal communications, 1-3
Terminal facilities, 7-1
 interactive terminal communications, 7-1
 network command terminals, 7-2
 Network virtual terminal (NVT), 7-3
 Phone, 7-1
 sample application, 9-13
 TLK, 7-2
Terminal server, 3-11
Terminating a logical link, 5-7
TLK, 7-2
 sample application, 9-14
Topology, 1-3
Trace, 11-2, 11-3
Transmission modes, 2-4, 10-10
Transmission speeds, 3-3
Transmit password, 10-6
Tributary addresses, 10-9

U

Up-line dump, 10-12, 11-2, 11-7
User layer, 1-5

V

Virtual Network Processor, see VNP

W

Wide area networks, 3-2
 examples, 3-4

X

X.25, 1-4, 3-12, 8-1
 access methods, 8-4
 circuits, 8-3
 definition, 8-1
 DTEs and DCEs, 8-2
 levels, 8-2
 native mode, 8-5
X.25/X.29 extension package (XEP), 8-1
 for VAX-11, 8-5, 8-6
X.28, 8-6, 8-7
 see also X.25
X.29, 8-1, 8-5, 8-7
 see also X.25
X.29 Virtual Terminal Driver, 8-6
 see also X.25
X.3, 8-5, 8-7
 see also X.25
XEP
 see X.25/X.29 extension package

3270 terminal emulation, 3-12, 8-9
 see also DECnet/SNA

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or
Country

— — —Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
1925 ANDOVER STREET TW/E07
TEWKSBURY, MASSACHUSETTS 01876

— — Do Not Tear - Fold Here and Tape —

Cut Along Dotted Line